Being a puppet master

More money, more time, more happiness, less work

Thomas Merkel
<tm@core.io>

http://www.flickr.com/photos/serenaseblu/5062987760/
Agenda

• Overview

• Organize the master
  • Modules
  • Resources types

• Syntax (if, else, switch, case, class)

• Example

• Tools (facter, hiera, mcollective)

• Demo and recommendation

• It’s working time now :-)

Where I want to be ... need more alcohol
Basic Overview

• Stop administrating your env. and start developing it

• Supports Linux, BSD, Solaris and Windows

• Re-usable code for managing software and configuration

• Provides a domain specific language to scripts (classes, conditions, selectors, variables, …)
Basic Overview

• Support >20 different package providers
• Support >10 different init frameworks
• Control whenever a service needs to be started or stopped
• Service could be notified to restart
Things you need to know

• Nodes - Machine to configure, identify by hostname
• Modules - Collection of classes and files
• Class - A collection of resources related to each other
• Resources - Things like packages, files, users, etc.
• Defines - A function-like construct for resources
High-Level Overview

- **Puppet Master**
  - Modules
  - Configuration

- **General Infrastructure**
  - Monitoring
  - DNS
  - Syslog
  - Webserver

- **Software Development Environment**
  - Development
  - Testing
  - Acceptance
  - Production

- **Puppet Agent / Clients**
Network Overview

- Client or server initiated synchronizations
- CA on the puppet master to sign client certificates to verify authentication
- Transmission of all data between a master & client are encrypted

Puppet Client

8140/TCP  
Client initiated  
Puppet agent -t

SSL

Puppet Master

8139/TCP  
Server initiated  
Puppet kick  
deprecated, use mcollective

Puppet Client
Organize the master
Example (puppetmaster)

• Location mostly on Linux `/etc/puppet`

• There are multiple ways to the holy grail

```
|-- fileserver.conf
|-- hieradata
|  |-- common.yaml
|-- manifests
|  |-- node.pp
|  |-- site.pp
|-- modules
|  |-- ssh
|     |-- manifests
|     |  |-- init.pp
|     |-- templates
|            |-- sshd_config.erb
|-- puppet.conf
|-- templates
```
Module structure

- Encapsulate a logical segment of an machines setup
- Thousands of modules exists

- manifests/
  - Tell the module how to work
- files/
  - Static files needed for development
- templates/
  - Dynamic ruby-based templates
- lib/
  - Relevant ruby-based libraries
Resources types

- files & directories
- users & groups
- services
- packages
- crontabs
- mount points
- nagios
- selinux
- ssh keys
- third party repositories (yum, apt, etc.)

Many many more :-) 

http://docs.puppetlabs.com/references/latest/type.html
Syntax
# single class
class ntp { ... }

# inherited class
class sftp inherits ssh { ... }

# scoped class
class ntp::base { ... }
Syntax - Resources

```
service { 'httpd':
    ensure => running,
    enable => true,
    hasstatus => true,
    hasrestart => true,
}
```
Syntax - if/else

```php
if ($environment == "production") {
    include powerdns
} else {
    include bind
}
```
Syntax - switch/case

case $operatingsystem { 
    Debian|Ubuntu: { 
        include nagios::debian 
    } 
    CentOS: { include nagios::centos } 
}
## /etc/puppet/manifests/site.pp - first file

```puppet
Exec {
  path => [ "/usr/local/bin","/usr/bin","/bin","/usr/local/sbin","/usr/sbin","/sbin","/opt/local/bin","/opt/local/sbin" ]
}
```

## import some config files

```puppet
import "common"
```

# auto-config files that are deployed by limeade or some other services contains important variables and config settings for some puppet-modules

```puppet
import "import/*.pp"
```

# all nodes that are static configured

```puppet
import "nodes"
```
## /etc/puppet/manifests/node.pp

## default node, deploy on all nodes
node default {
  include sudo
  include concat::setup
  include apt
}

## qwecompany nodes
node qwecompany inherits default {
  include ssh
  include qwecompany_base
  include munin
}

node 'net-dev.qwe123.de' inherits qwecompany {
  include qwecompany_net
}
class ssh ($permitRootLogin='no', $port='22', $passwordAuth='no') {
  package {'openssh-server':
    ensure => present,
  }
  file {'/etc/ssh/sshd_config':
    content => template('ssh/sshd_config.erb'),
    mode => '0400',
    notify => Service['sshd'],
    require => Package['openssh-server'],
  }
  service {'sshd':
    name => 'ssh',
    ensure => running,
    enable => true,
    hasstatus => true,
    hasrestart => true,
    require => File['/etc/ssh/sshd_config'],
  }
}
### /etc/puppet/modules/ssh/templates/sshd_config.erb

Port `<%= port %>`

#Port 22

Protocol 2

![...]

![...]

# Authentication:

LoginGraceTime 2m

PermitRootLogin `<%= permitRootLogin %>`

StrictModes yes

#MaxAuthTries 6

![...]

![...]

Port `<%= port %>`

#Port 22

Protocol 2

![...]

![...]

# Authentication:

LoginGraceTime 2m

PermitRootLogin `<%= permitRootLogin %>`

StrictModes yes

#MaxAuthTries 6

![...]

![...]

# Authentication:
Facter

- Describes aspect of your machine - „facts“
- Facts written in Ruby
- Nice libraries of existing facts
- Custom facts are easy
tmerkel@arena:~$ facter
architecture => amd64
augeasversion => 1.1.0
domain => srv.avira.net
facterversion => 1.7.5
filesystems => ext3, ext4, vfat
fqdn => arena.srv.avira.net
hardwareisa => x86_64
hardwaremodel => x86_64
hostname => arena
interfaces => eth0, eth1, lo
ipaddress => 62.146.210.70
ipaddress_eth0 => 62.146.210.70
ipaddress_eth1 => 62.146.211.70
ipaddress_lo => 127.0.0.1
is_virtual => true
Hiera

- Hierarchical data lookup system
- Structured data backend
  - YAML, JSON, current puppet state
- Example: storage ssh keys in YAML hiera db
Hiera - default lookup

- Default lookup for class parameter

```yaml
# /etc/puppet/hieradata/web01.example.com.yaml
---
ssh::permitRootLogin: "yes"
ssh::port: 22

# /etc/puppet/hieradata/common.yaml
---
ssh::permitRootLogin: "no"
```
Hiera - lookup function

```yaml
# /etc/puppet/hieradata/appservers.yaml
---
proxies:
- hostname: lb01.example.com
  ipaddress: 192.168.22.21
- hostname: lb02.example.com
  ipaddress: 192.168.22.28
```

```perl
# Get the structured data:
$proxies = hiera('proxies')

# Index into the structure:
$use_ip = $proxies[1]['ipaddress'] # will be 192.168.22.28
```
Marionette Collective

http://www.flickr.com/photos/leonardo_pilara/12121189764/
MCollective

- Manage / Control / Execute
  - Services
  - Packages
  - Process information
  - Facter facts
  - Pings
Demonstration
Recommendation

• Use `git`, bitch :-)  
  • git for every puppet module  
  • git submodules to combine them  
  • Minimum number of puppet master (it can handle >5000 servers without any problem)  
  • Manage everything with puppet, don’t make exceptions on an server
Recommendation

• Scale the master with unicorn or some other ruby thingy

• Start using it, if something fails create a new puppet master and move modules

• Check out public modules that are available
  • https://github.com/drscream

• Please test the puppet agent on Windows

• Check mcollective if the usage would be helpful
THE END

What’s next?
It’s working time :-)  

• Puppet master
  • root@puppet.qwe123.de

• Puppet clients
  • root@client01.qwe123.de
  • root@client02.qwe123.de
• missing windows server