Puppet: System Configuration Management quick overview

François Deppierraz
francois@ctrlaltdel.ch

Swiss Puppet User Group, 1st Meeting

February 18th 2010

Why System Configuration Management ?

- Could a computer infrastructure ever converge ?
  - Certainly not !
  - So you have to find a way to slow down divergence
- Different types of configurations
  - Different purposes
  - Different hardware
  - Different networks
- Multiple System Administrators working in team
- "Uh, I can't remember which hack I had to make to get this service running !"

What is Puppet ?

- Puppet is system administration - Automated
  - Administer One Server or 1,000
  - Configuration is OS/distribution independant
  - Repeatable configurations
- Developped by an active developper community
- Project started in 2005
- GPL
- Written in Ruby
- Portable
  - Linux
  - *BSD
  - Solaris
  - MacOS X
  - Windows (! ?)

What makes Puppet so cool ?

- cfengine done (a bit more) right
- New server up and running in less than 10 minutes
  - Well, usually...
- Type based abstraction layer
  - file
  - user
  - package
  - service
  - cron
- Extensible
  - new types
  - node classification
- Active project and community

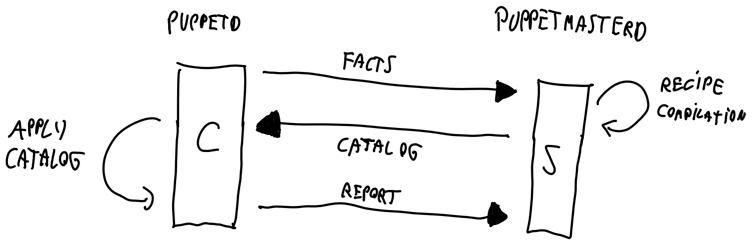
Puppet types

- Each type can have multiple providers
- package
  - dpkg
  - rpm
  - yum
  - apt-get
  - portage
  - and so on...
- user
  - netinfo
  - pw
  - useradd

Facter

- "A cross-platform Ruby library for retrieving facts from operating systems"
- Extensible, easy to add new facts
- Allows puppet definition modification based on client facts

How does it work ?



How does it work ? (2)

- 2 operation modes
  - Interpreter mode
  - Client-Server mode
- Client-Server mode
  - Client send a definition request with facts
  - Definition is compiled on server
  - Sent to client
  - Abstract types $- >$ Real configuration (providers)
  - Current state saved on client

Node and Classes

- A node is any client identified by its hostname
- A class can include classes
- A node can include classes
  - Called node classification
  - Using LDAP
  - Using custom scripts
- Classes support inheritance
- Example : apache
  - class apache
    - class apache : :ssl inherits apache
    - class apache : :snvserver inherits apache

User configuration

```
file {"/etc/passwd":
  ensure => present,
  owner  => root,
  group  => root,
  mode   => 644
}

user {"francois":
  ensure   => present,
  password => "...",
  uid      => 1000,
  groups   => [adm]
}

user {"guest":
  ensure => absent
}
```

Apache configuration

```
package {"apache":
  ensure => installed
}

service {"apache":
  ensure => running
}

vhost {"www.foobar.com":
  docroot => "/var/www/www.foobar.com/htdocs",
  aliases => ["foobar.com", "barfoo.com"],
}

vhost {"www.test.com":
  docroot => "/var/www/www.test.com/htdocs",
  aliases => ["test.com"],
}
```

Request Tracker

```
node 'rt.nimag.net' {
  include base
  include apache
  include postgresql
  include rt
}
```

Any Questions ?

François Deppierraz
francois@ctrlaltdel.ch