



Getting started with Puppet

Jeremy MATHEVET
12/09/2012

Topics

- **Configuration management**
- **Puppet Installation**
- **Puppetmaster**
- **Puppet language basics**
- **Puppet language advanced**

Configuration management

Puppet, Chef, Cfengine

Definition

Configuration Management is the process of standardizing resource configurations and enforcing their state across IT infrastructure in an automated yet agile manner.

— **Puppet Labs**

Why do we need it

- Heterogeneous Operating Systems
- Servers profusion
- Increasing deployment needs
 - Cloud



Different solutions

3 ways for deploying software to the SI

Manually 

- Could take very long time

By scripts 

- Could be quite tricky
 - Multi-environment case

By configuration management software 

Configuration management software

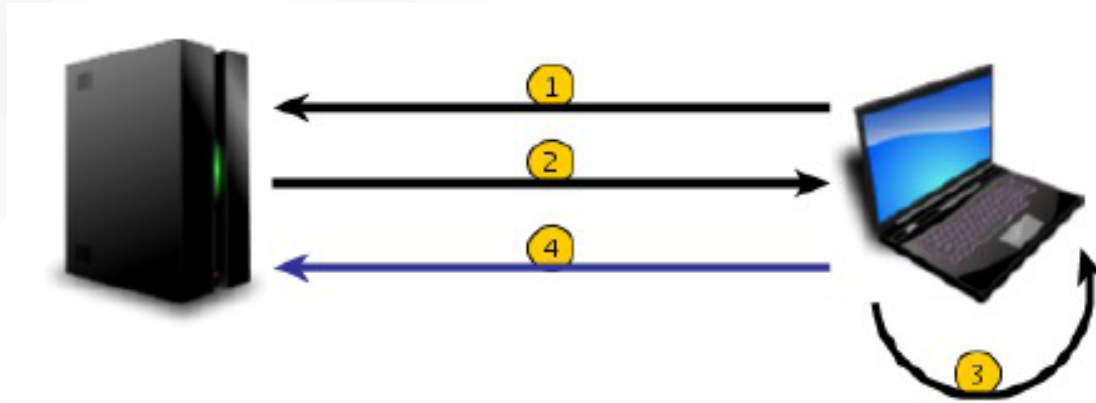
Pros :

- Centralized management
- Automated management
- Mass deployment
- Configuration customization
- Abstraction layer
- Idempotence
- "DevOps ready"

Principle

- A client/server architecture.
- The server has a reference configuration.
- The client queries the server.
- The client makes change in order to match the reference configuration.

Principle

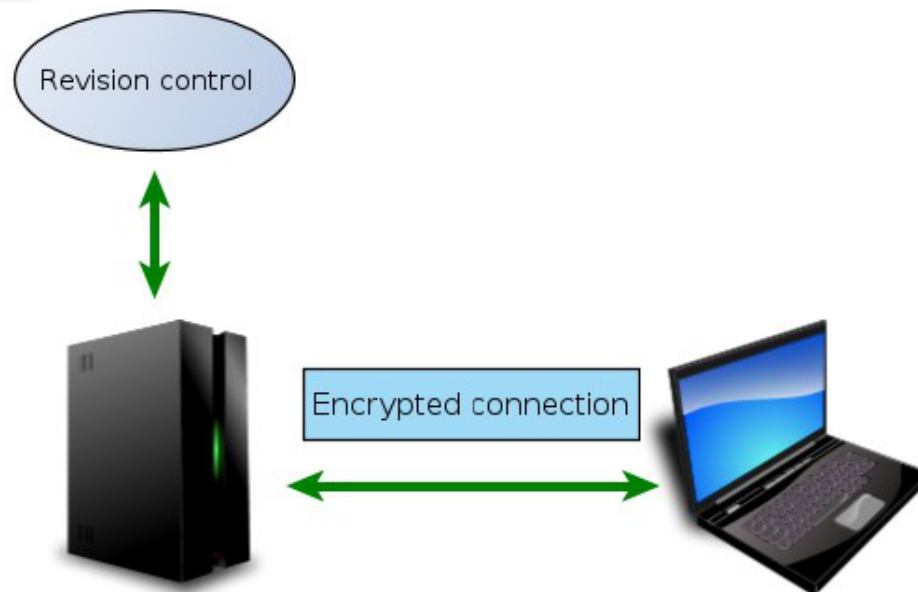


1. Pull Request
2. Reference configuration
3. Operations
4. (optionnal) Message code / error

Principle

Version control and security

- Revision control used to keep a trace of config revisions
 - Not mandatory but hugely recommended
- Encrypted connection



Principle

What you can do

- File transfer
- File edition
- Service management
- Package management
- Mount points management
- Cron management
- VLAN management
- Command launching
- ...

Comparison

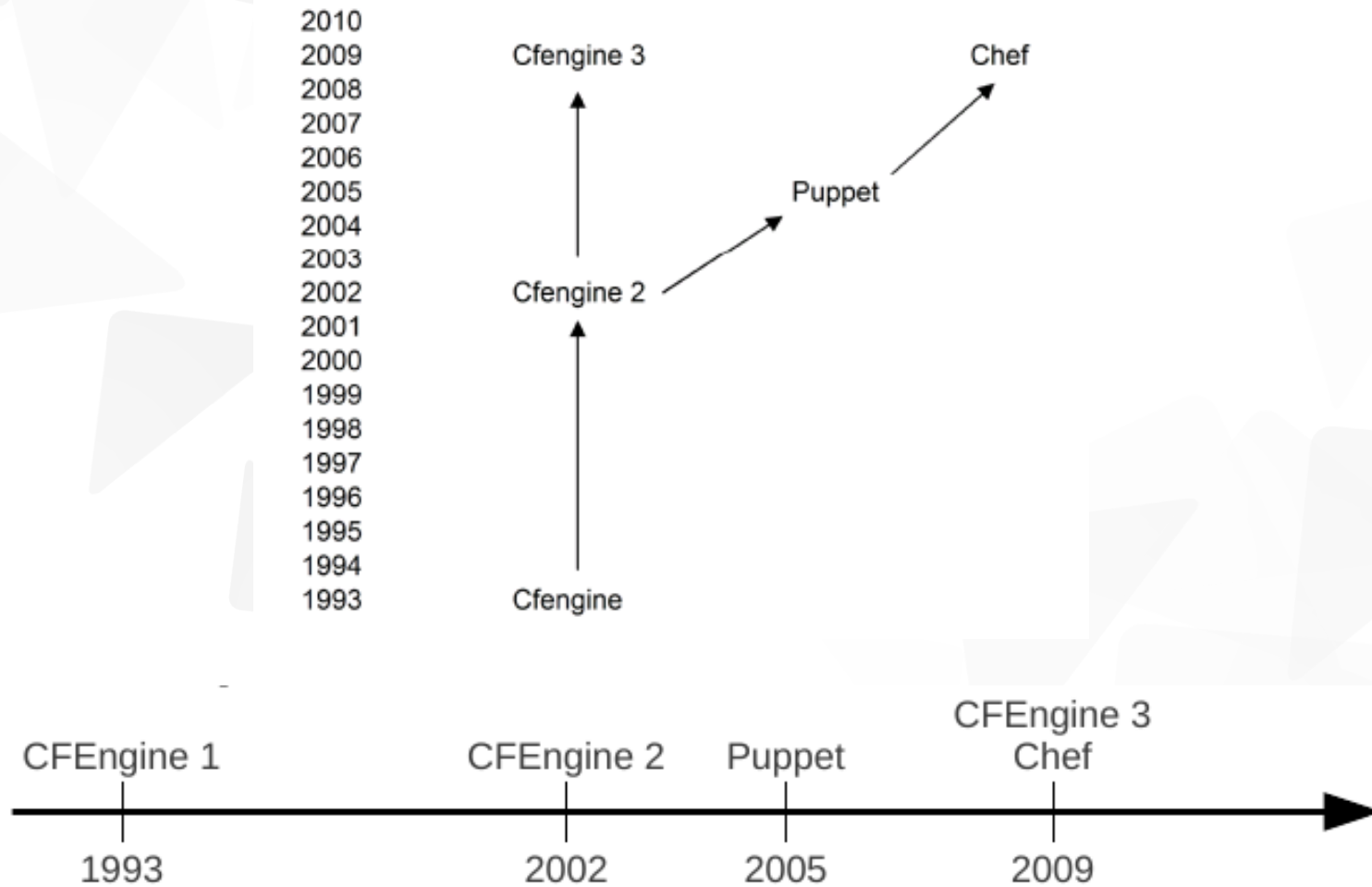
3 major solutions :

- Puppet
 - The most popular
 - Easiest solution
- Chef
 - Puppet fork
 - Still young but very promising
- CFEngine
 - The first CM software
 - The most resource effective
 - More complex



Comparison

Timeline :



Comparison

Some differences



Technology	Ruby	Ruby	C
Configuration	Dedicated language	Ruby	Dedicated language
Configuration method	Plain text files	Plain text files CLI tool Web UI	Plain text files
Licence	Apache	Apache	GPL

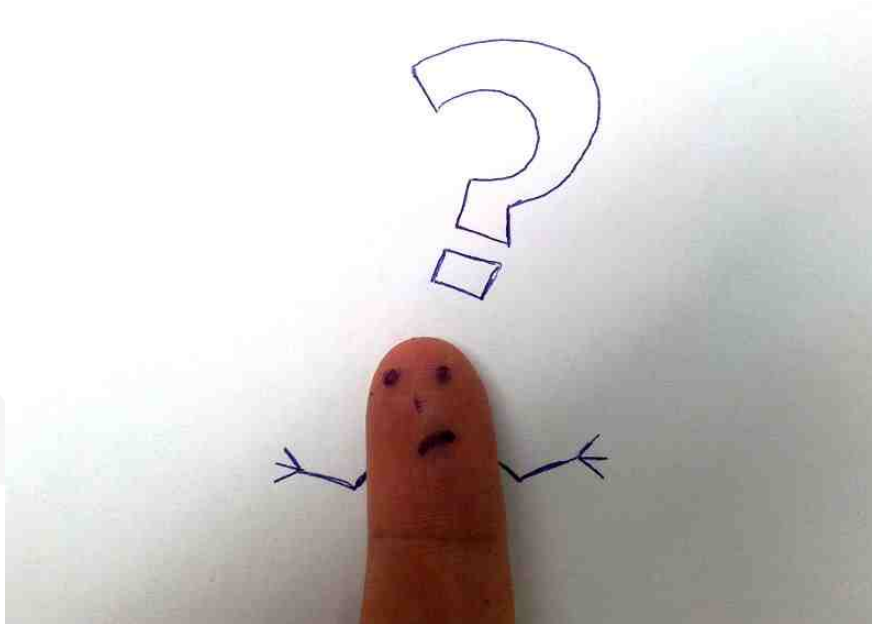
Comparison

Some similarities

- Same origin
- Specially designed for config management
- Client/server model
 - Standalone mode possible
- CLI interface
- Need to increase your competence



Do you have any questions ?



Puppet Installation

Install && setup

Puppet

Some details and features

- Created in 2006 by Puppet Labs
- Latest version : 3.0.0
- The easiest solution
- Dedicated declarative language
 - Inspired by Ruby
- Modular configuration
- System profiling with Facter
- Template support
- Asymmetric Key Encryption

Puppet

Prerequisite and sources

- Prerequisite
 - Configured DNS
 - Ruby
- Installation sources :
 - Distribution repositories
 - RubyGem
 - Sources

Puppet

Client and server

Puppet server : Puppetmaster

Puppet client : Puppet (agent)

Main steps once installed :

- Puppet agent installation on client hosts (also called a node).
 - Enter the server FQDN into */etc/puppet/puppet.conf*
 - The agent checks every 30 mn by default
- Certificate generation
 - CSR for the node generated on the puppetmaster during the first agent request

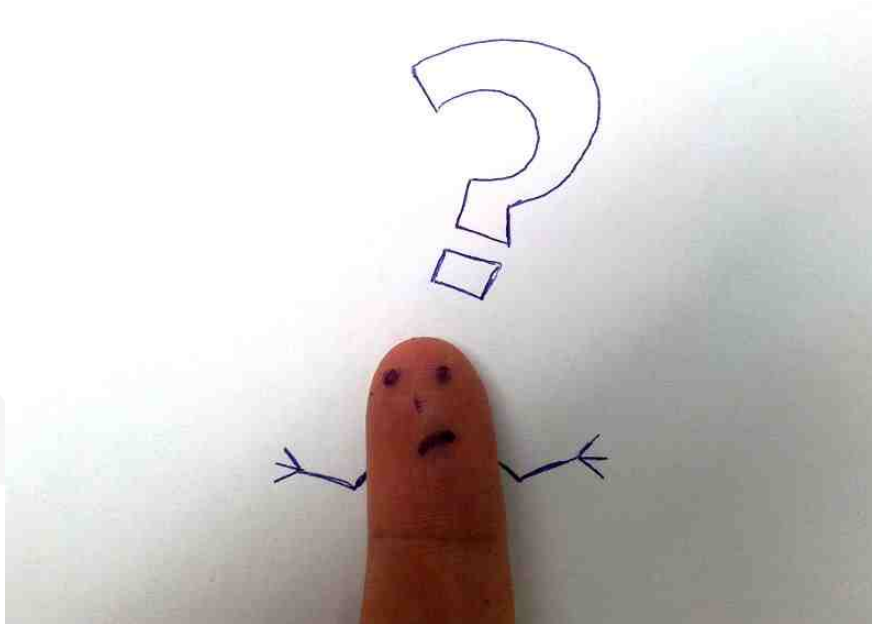
Working with Puppetmaster

Working with Puppet CA

```
[root@puppetmaster ~]# puppetca --sign fqdn.utopia.net
```

Options	Definitions
--sign	Sign the certificate request for the following host
--list	List certificate request
--all	Used with --sign, sign all certificate request
--clean	Remove all files related to the host from the CA
--print	Print the full text version of a host's certificate
--revoke	Remove all files related to the host from the CA

Do you have any questions ?



Puppet

Puppetmaster

Puppetmaster said...

One important thing

Infrastructure is code.

Definition

Manifest – Puppet programs are called « manifests ». They use the .pp file extension. These programs contain one or more class.

Puppet configuration tree

Configuration files overview 1/2

- puppet.conf
 - General Puppetmaster settings
- auth.conf
 - General ACL
- fileserver.conf
 - ACL for the Puppet fileserver
- manifests directory
 - site.pp : global defaults configuration
 - nodes.pp : manage hosts

```
server:/etc/puppet$ tree
.
├── auth.conf
├── fileserver.conf
├── manifests
│   ├── nodes.pp
│   └── site.pp
├── modules
│   └── test
│       ├── files
│       │   └── test.txt
│       ├── manifests
│       │   └── init.pp
├── puppet.conf
└── templates
```

Puppet configuration tree

Configuration files overview 2/2

- modules directory
 - Contain one subdirectory per module
 - manifests/init.pp
 - The module code
 - The best way to use Puppet
- templates directory
 - The default directory which contains templates.

```
server:/etc/puppet$ tree
.
├── auth.conf
├── fileserver.conf
├── manifests
│   ├── nodes.pp
│   └── site.pp
├── modules
│   └── test
│       ├── files
│       │   └── test.txt
│       ├── manifests
│       │   └── init.pp
├── puppet.conf
└── templates
```

Puppet configuration files

puppet.conf

```
[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
templatedir=$confdir/templates
server=debiantest.decilex.net
downcasefacts = true
```

Puppet configuration files

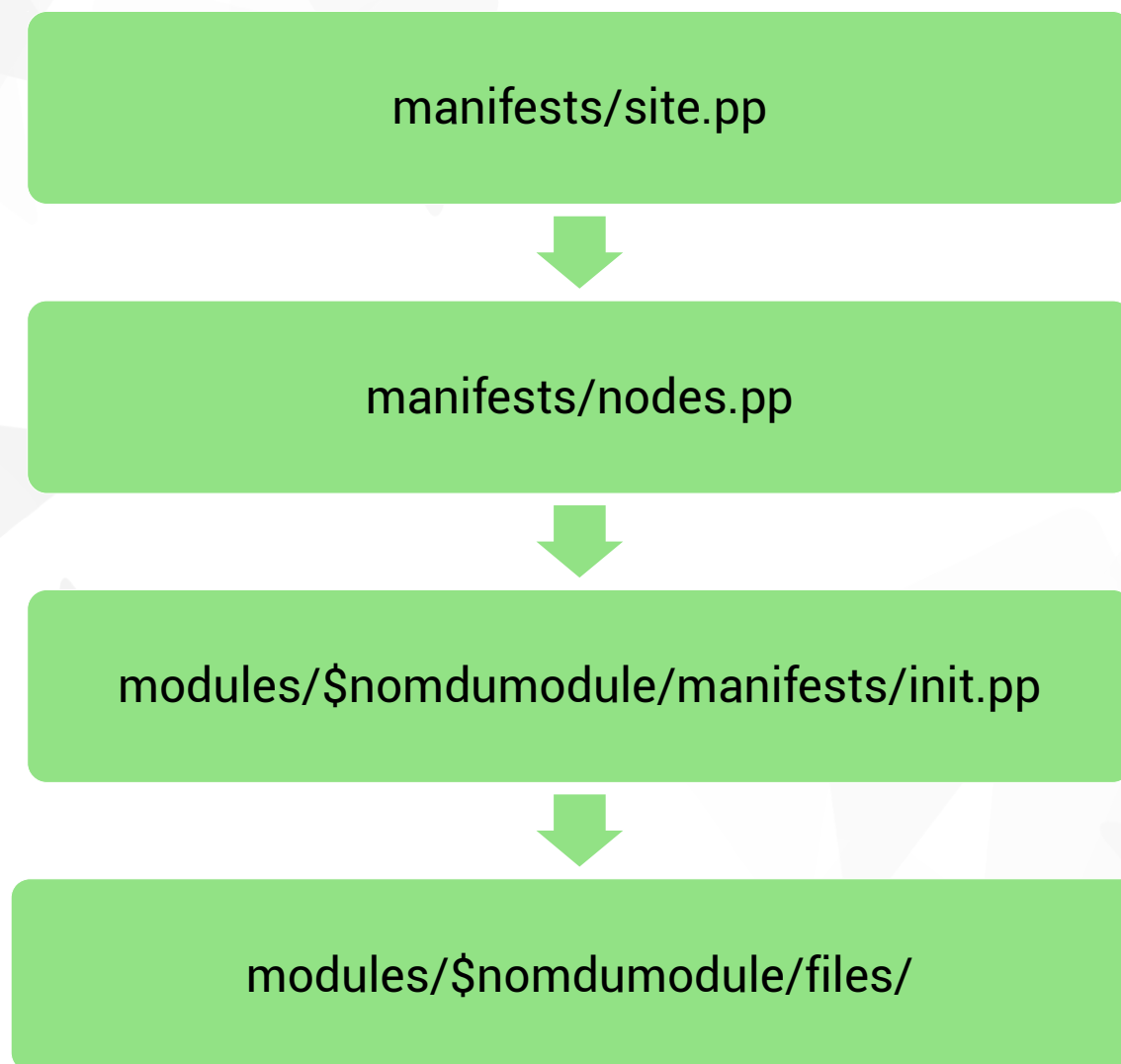
manifests/nodes.pp

- Skeleton :
 - node 'fqdn' { include module}
- Node declaration
- Module inclusion
 - Inheritance

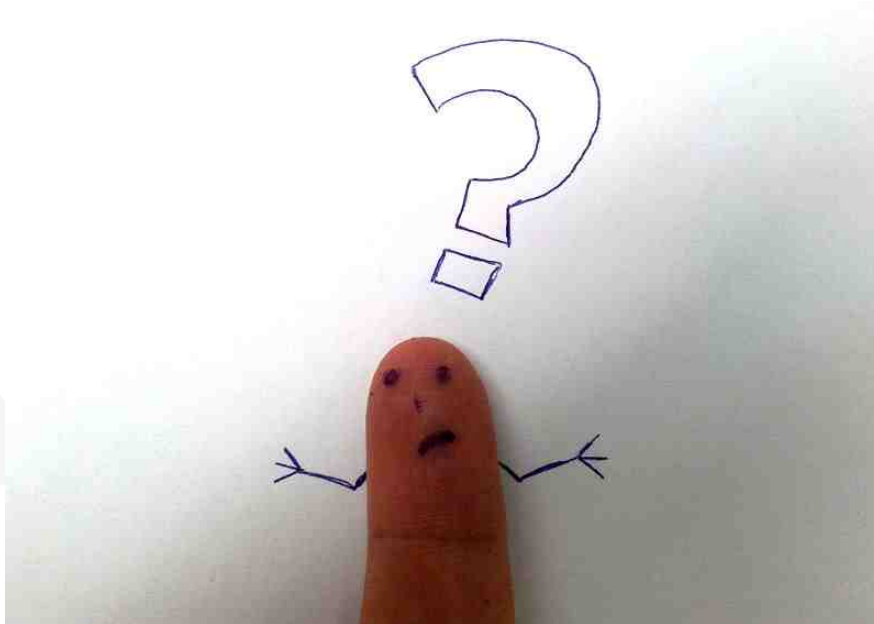
```
node basenode {  
  include ssh  
  include ntp  
}  
  
node 'test.utopia.net' inherits basenode {  
  include nagios  
}
```

Puppet configuration files

Reading order



Do you have any questions ?



Puppet language basics

Puppet, Chef, Cfengine

The declarative language

About the language

- With Puppet, we declare how the node must be.
- Everything you want to manage have to be explicitly declared.
- A Puppet program is called a manifest
 - Central manifest : site.pp
 - Puppet load modules manifests
- Into manifests, we define classes.
 - We write resources inside these classes

The declarative language

Resources

- The fundamental unit of modeling
- Like a “function”
 - Inside, a series of attributes and their values
- Resources types and attributes are predefined by Puppet
- List of available resources
 - <http://docs.puppetlabs.com/references/stable/type.html>
- Skeleton
 - Ressource-name { 'title' : attribute = value }

Resources

File

- Manage files
 - Content
 - Permissions
 - Ownership
 -
- Source attribute
 - Copy a file from the Puppetmaster to the node
 - puppet:/// followed by the relative source of the file placed in /etc/puppet/modules/module-name/files/

```
file { '/etc/passwd':  
  owner => 'root',  
  group => 'root',  
  mode  => '0644',  
  source => 'puppet:///base/passwd'  
}
```

Resources

Package

- Manage packages
 - Wide provider support
 - APT
 - Aptitude
 - YUM
 - And more..
 - Install, upgrade, uninstall packages
 - The last or defined package version

```
package { 'openssh-server':  
  ensure => installed  
}
```

```
package { 'openssh-server':  
  ensure => latest  
}
```

```
package { 'openssh-server':  
  ensure => absent  
}
```

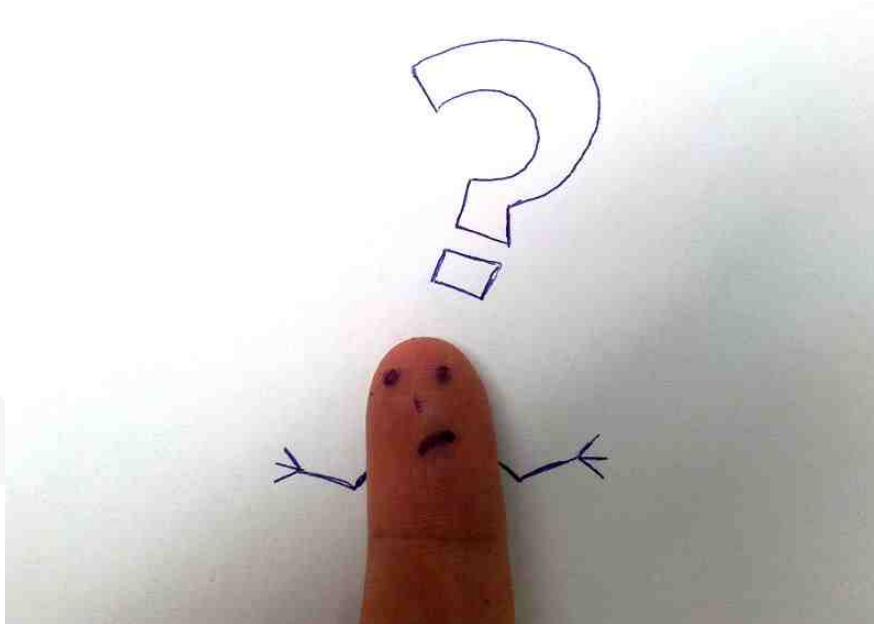
Resources

Service

- Manage services
 - Wide provider support
 - Init
 - Systemd
 - Upstart
 - And more...
 - Start, stop, restart, start on boot (enable) services

```
service { 'snmpd':  
  ensure => running,  
  enable => true,  
  hasrestart => true,  
  hasstatus => true  
}
```

Do you have any questions ?



Puppet language advanced

Dive into Puppet

Facter

The system profiler

- Software used by Puppet
- Installed on nodes
- Collect various data, "facts", on node
- Many facts already defined by Facter
 - Possibility to create your own facts

```
~ # facter
architecture => i386
domain => utopia.net
facterversion => 1.5.7
fqdn => debiantest.utopia.net
hardwaremodel => i686
hostname => debiantest
id => root
[...]
```

Variables

Variables into classes

- Begin by \$
- Can use facts or you own defined variables
- Often used with conditional statements
 - Case statement
 - If statement

```
case $operatingsystem {  
  centos, redhat: { $service_name = 'ntpd' }  
  debian, ubuntu: { $service_name = 'ntp' }  
}  
  
service { 'ntp':  
  name    => $service_name,  
  ensure  => running,  
  enable  => true  
}
```

Conditional statements

If/Elsif/Else Statement

- Based on
 - the truth value of a variable
 - the value of an expression
 - The truth of an arithmetic expression

```
if $variable {  
  file { '/some/file': ensure => present }  
} else {  
  file { '/some/other/file': ensure => present }  
}
```

```
if $server == 'mongrel' {  
  include mongrel  
} elsif $server == 'nginx' {  
  include nginx  
} else {  
  include thin  
}
```

```
if $ram > 1024 {  
  $maxclient = 500  
}
```

Expressions

Operators

- Operators usable in *if* statements
 - Boolean expressions
 - And, or, not
 - Comparison expressions
 - == != < > <= > >=
 - Arithmetic expressions
 - + - / * >> (left shift) << (right shift)
 - Regular expressions
 - =~ !~
 - “in” expressions
 - allows to find if the left operand is in the right one.

Templates

Personalized text files

- Permit to have personalized configuration per node
 - Use ERB language
 - Retrieve and use facts
 - Use file resource
 - ERB file placed in module template directory

```
file {"hosts":  
  path  => "/etc/hosts",  
  owner  => root,  
  group  => root,  
  mode   => 644,  
  content => template("base/hosts.erb")  
}
```

```
<%= ipaddress %> <%fqdn> <%hostname>
```

Resources relationship

Relationship meta-parameters

- **before**
 - Resource is applied before the target resource
- **require**
 - Resource is applied after the target resource
- **notify**
 - Like before + The target resource will refresh if the notifying resource changes
- **subscribe**
 - Like require + The subscribing resource will refresh if the target resource changes.

Resources relationship

Ordering relationship

```
package { 'openssh-server':  
  ensure => present,  
  before => File['/etc/ssh/sshd_config'],  
}
```

```
file { '/etc/ssh/sshd_config':  
  ensure => file,  
  mode   => 600,  
  source => 'puppet:///modules/sshd/sshd_config',  
  require => Package['openssh-server'],  
}
```

These two examples are mutually-exclusive

Resources relationship

Notification relationship

```
file { '/etc/ssh/sshd_config':  
  ensure => file,  
  mode   => 600,  
  source => 'puppet:///modules/sshd/sshd_config',  
  notify => Service['sshd']  
}
```

```
service { 'sshd':  
  ensure  => running,  
  enable  => true,  
  subscribe => File['/etc/ssh/sshd_config']  
}
```

These two examples are mutually-exclusive

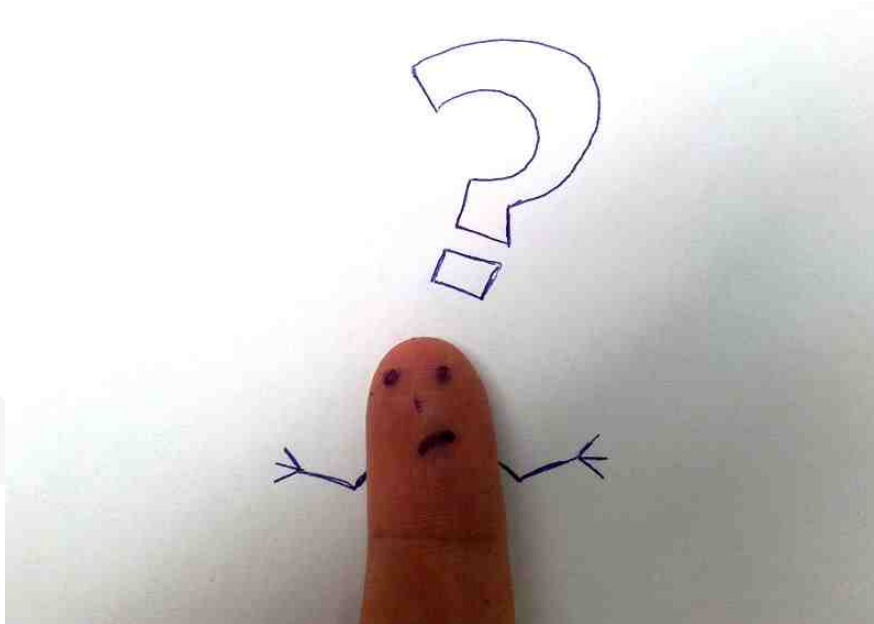
Resources relationship

Chaining and refreshing

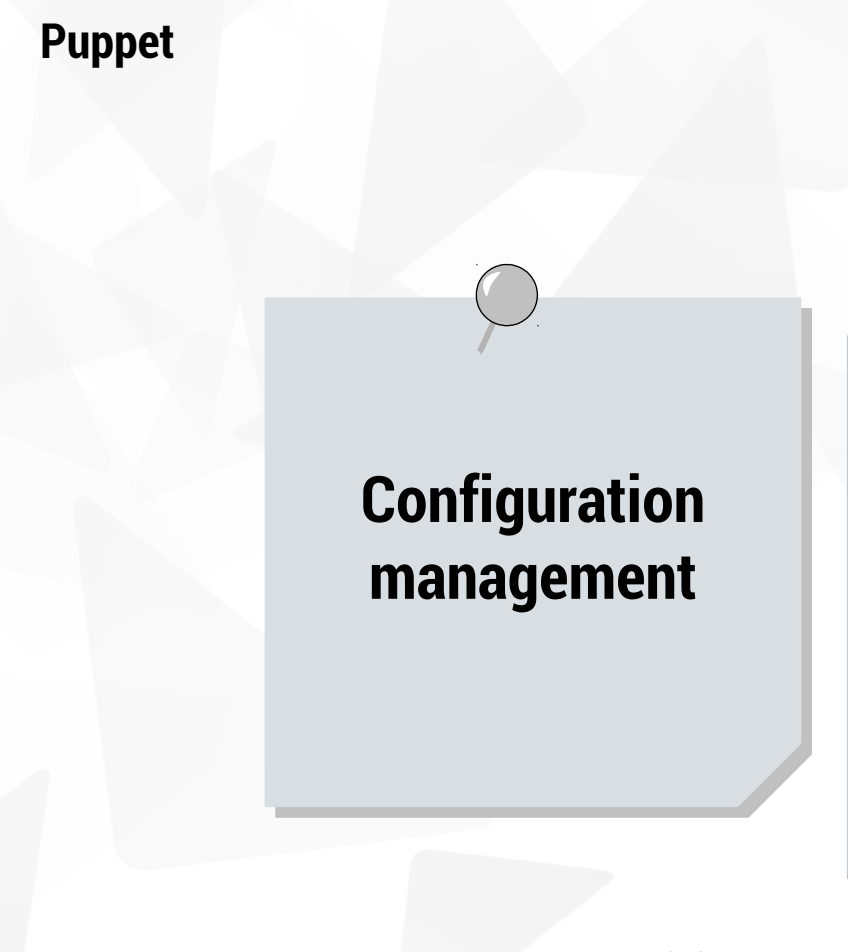
- Ordering resources
 - The resource on the left is applied before the resource on the right.
 - ->
- Refreshing
 - Kind of trigger
 - Restart a service after a file update
 - ~>

```
# ntp.conf is applied first, and will notify the ntpd service if it changes:  
File['/etc/ntp.conf'] ~> Service['ntpd']
```

Do you have any questions ?



Summary



**Configuration
management**

Puppet installation

Puppetmaster

**Puppet language
basics**

**Puppet Language
advanced**



Contact

Jeremy MATHEVET
@Jeyg