RAISIN

Réseau Aquitain des Informaticiens Systèmes INter-établissement

GESTION D'INFRASTRUCTURE AVEC PUPPET

Séminaire RAISIN du 27/05/2010

Florent Paillot & Christophe Delalande

INRIA Bordeaux - Sud-Ouest

Sommaire

- Qu'est ce que Puppet ?
- Pourquoi faire
- Avantages/Inconvénients
- Architecture
- Installation
- Fonctionnement
- Organisation
- Présentation du langage
- Bonnes pratiques
- Bilan

Qu’est ce que Puppet ?

- Un outil de gestion automatique de configuration
  - Amener et maintenir un système dans un état désiré (rôle) à partir association nom de machine/recette(s)
- Basé sur une architecture client/serveur
- Multi-plateformes (Linux, Unix, Windows en préparation)
- Écrit en Ruby
- Utilise un langage déclaratif
- Développé par Luke Kanies et PuppetLabs sous GPLv2
- Concurrents: cfengine, bcfg2, lcfg, chef

Pourquoi faire (1) ?

- Eviter les tâches répétitives : on ne fait le travail de configuration qu'une seule fois
- Permettre une configuration homogène de parc
- S'assurer que les machines conservent leur configuration

Pourquoi faire (2) ?

- Déploiement rapide de machine
- Sécurité
  - Plus facile de déployer massivement un durcissement
  - Écart par rapport à la cible ➔ reporting et remise en état
- Brique d'une base de gestion des configurations (CMDB ITIL)

Puppet / Pro (1)

- Orienté administration système
  - Couche d'abstraction de ressources
    - services, paquets, fichiers, utilisateurs, cron, …
  - Facter
- Configuration puissante (tests conditionnels, dépendance, héritage, template, …)
- Code compact et lisible

Puppet / Pro (2)

- L'indisponibilité du serveur n'est pas bloquante
- Rentrer/sortir dans le système Puppet = installer/supprimer le daemon client
- Extensible : on peut commencer par ne gérer que quelques paramètres/fichiers, puis accroître la surface progressivement.
- Développement et communauté actifs

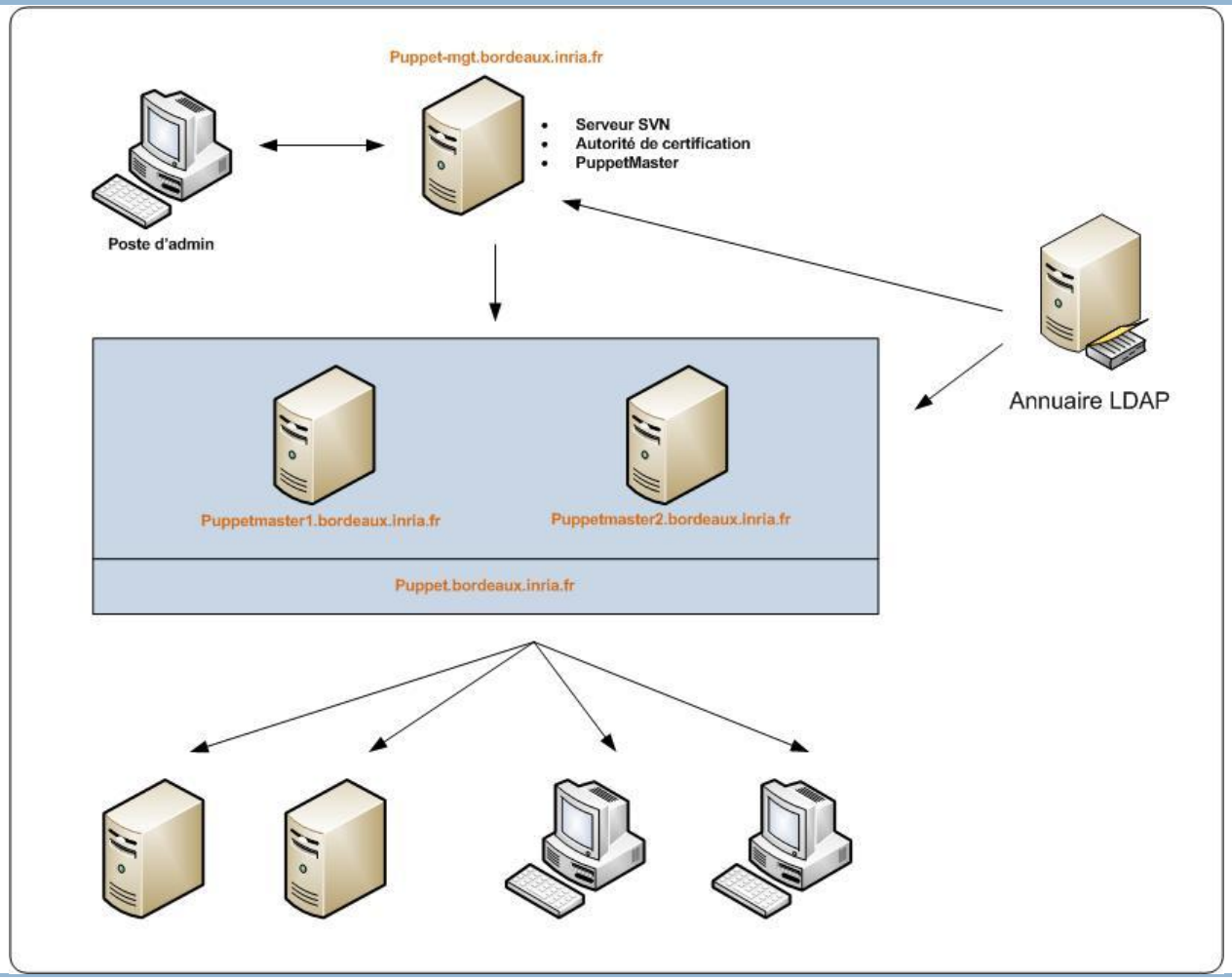
Puppet / Cons

- Ticket d'entrée
- Pas forcément adapté à tous les services
- Intérêt faible en dessous d'une taille critique
- Permet de se tirer une ~~balle~~ rafale dans le pied

Architecture

- Serveur : puppetmasterd
  - TCP/8140
  - Utilisateur "puppet"
  - Communications chiffrées
  - PKI intégrée
  - Serveur de fichiers
- Client : puppetd
  - Se connecte au serveur toutes les 30 mn
  - Utilisateur "root"

Architecture



Composants

- facter
  - S'exécute sur le client
  - Renvoi couple attribut/valeur => facts

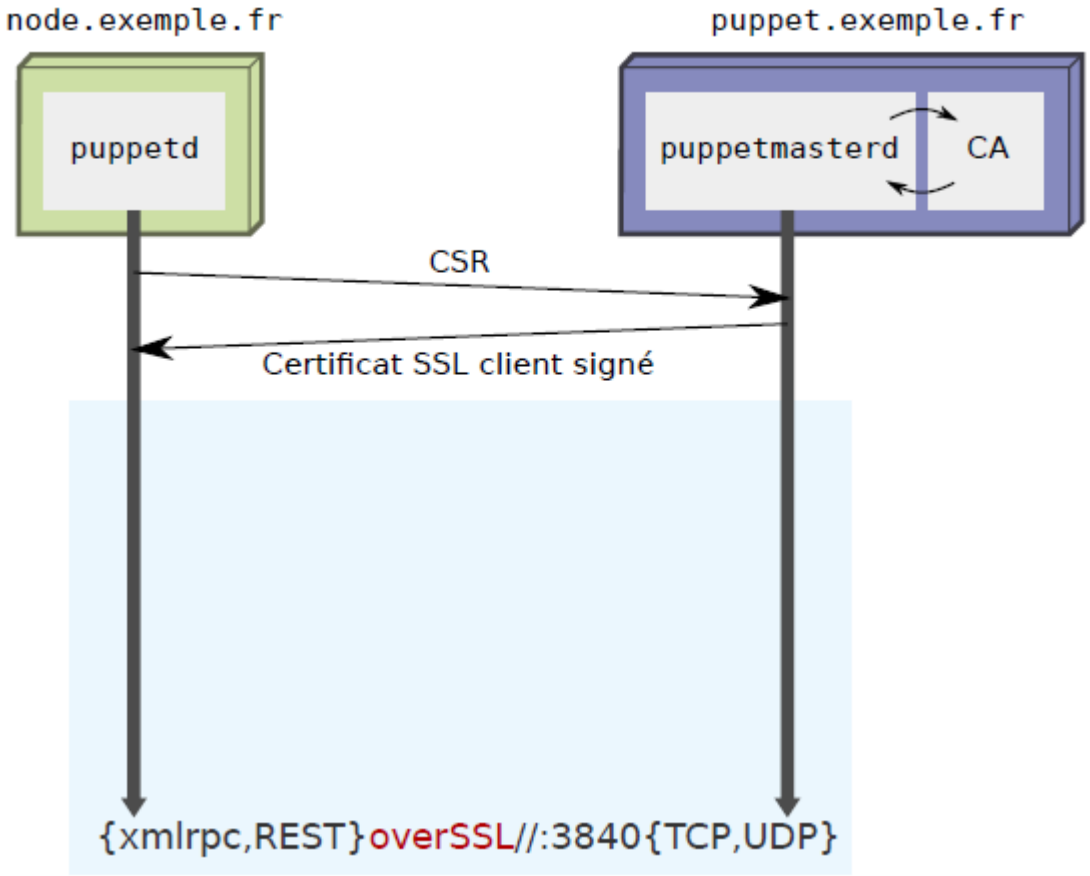
```
architecture => x86_64
fqdn => puppet.bordeaux.inria.fr
hostname => puppet
operatingsystem => CentOS
operatingsystemrelease => 5.5
kernelversion => 2.6.18
```

- puppetca
  - Utilitaire pour approuver les certificats clients

PKI

- Puppet utilise une PKI pour sécuriser les échanges entre le serveur et le client
- Les clients demandent la signature de leur certificat
- Possibilité d'utiliser une liste de révocation

PKI: schéma


node.exemple.fr
puppet.exemple.fr
puppetd
puppetmasterd
CA
CSR
Certificat SSL client signé
{xmlrpc,REST}overSSL//:3840{TCP,UDP}

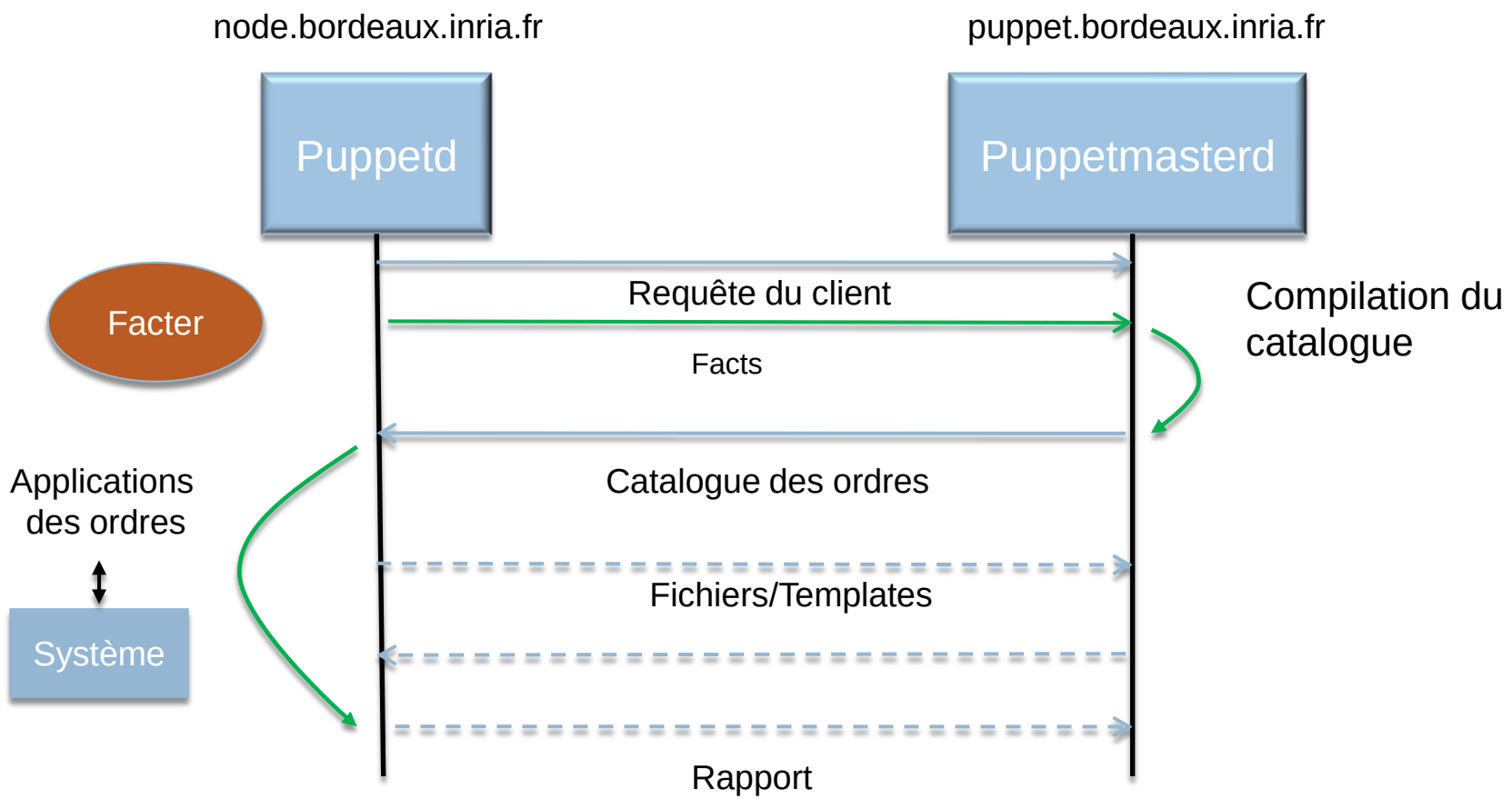
Installation

- Fedora/CentOS/RedHat
  - yum install puppet puppetmaster
  - Utiliser le dépôt EPEL
- Debian/Ubuntu
  - apt-get install puppet puppetmaster
- Ruby gem
  - gem install puppet -y

Commandes

- Démon puppet
  - /etc/init.d/puppet(master) start|stop|restart
- puppetca
  - (client)# puppetd --server <server-name> --waitforcert 60 --test --d
  - (serveur)# puppetca --list
  - (serveur)# puppetca --sign <client-name>
- Debug
  - puppet(master)d --test -d -v

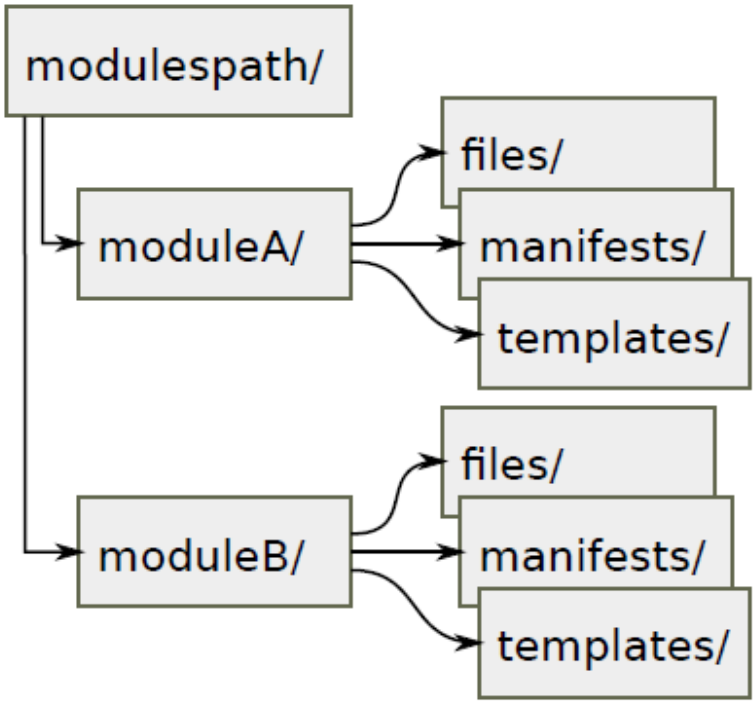
Fonctionnement de Puppet



Organisation des fichiers

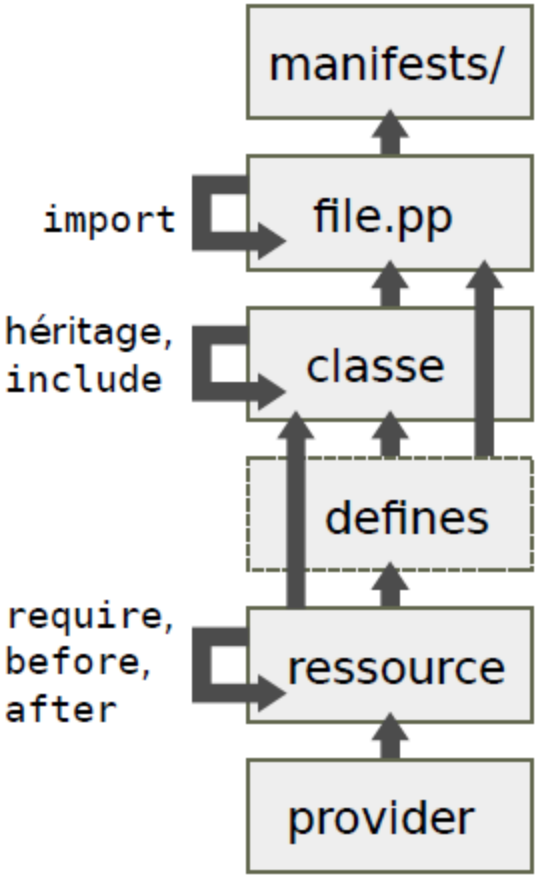
- Configuration
  - /etc/puppet/puppet.conf (client & serveur)
- Cœur du système
  - /etc/puppet/manifest/site.pp (serveur)

Modules



- Un **module** est **un ensemble logique** de composants dédiés à un sous-système particulier
- 3 types de composants:
  - Des **manifests** (*.pp) écrits en "langage Puppet"
  - Des **fichiers** de configuration (ex: sshd_config)
  - Des **templates** (*.erb)
- Conçus pour être portables

Les manifests



- Le dossier manifests/ contient un ou plusieurs fichiers manisfest (extension .pp)
- Le fichier init.pp est le premier fichier traité. Il peut appeler d'autres fichier via **import**

Le langage Puppet

- Les concepts
  - Ressources
  - Classes
  - Variables
  - Nodes
  - Définitions
  - Templates

Ressources

- Un élément que Puppet sait configurer
  - file (contenu, permissions, propriétaire)
  - package (présence ou absence)
  - service (activation/désactivation, démarrage/arrêt)
  - exec (exécution commande)
  - Et aussi cron, group, host, user, yumrepo, mount, sshkey…

Ressources: exemple

Type

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

file { "/etc/ssh/sshd_config":
    path    => "/etc/ssh/sshd_config",
    owner   => root,
    group   => root,
    mode    => 700,
    require => Package["openssh-server"],
    source  => ["puppet:///ssh/sshd_config.${hostname}"],
    }
```

Classes

- Un ensemble nommé de ressources
- Instance unique pour un hôte
- Peut inclure ou hériter d'autres classes

```
class ssh::server {
        include linuxserver
        package { "openssh-server":
                ensure => installed
        }
         …
}
```

Classes : héritage

- Permet d'hériter des valeurs définies dans la classe parente
- Intéressant pour modifier des attributs de la classe parente ou pour surcharger ses attributs
- Sinon, préférer l'inclusion

```
class unix  {
    file ["etc/passwd","/etc/shadow"] { group => root }
   file {"etc/group" :
         group => root }
}

class freebsd inherits unix {
    File["/etc/passwd","/etc/shadow"] { group => wheel }
}
```

Variables

- Possède une valeur valable dans une espace de nom
- Surcharge

```
$server = « primary »

Class master {
        file {« /etc/serveur.conf »:
                content => « $server »,
                ensure => present,}
}
Class slave {
        $server= « secondary »
        include master
}
```

Nodes

- Un bloc de configurations applicable à un client
- Peut contenir des ressources, définitions, classes

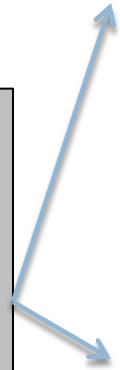
```
node ‘ldap1.inria.fr' {
        $log_type = "local"
        $puppet_env = "production"
         include server::debian
         include server::debian::ldap

                  }
}
```

Définitions

- Ensemble de ressources
- Pouvant avoir plusieurs instances pour un même hôte
- Ex: virtualhost apache

Définitions: exemple


```
define apache::vhost-ssl (
        $ensure=present,
        $ip_address="*",
        $cert=false,
        $certkey=false,
        $cacert=false,
        $certchain=false,
        $sslonly=false,
)
```



```
node 'srvweb1.bordeaux.inria.fr' {

include apache
apache::vhost-ssl { « site1.bordeaux.inria.fr":
                ensure => present,
                ip_address => « X.X.X.X",
                cert => "puppet:///apache/ssl-certs/site1.inria.fr.pem",
                certkey => "puppet:///apache/ssl-certs/site1.inria.fr.key",
                cacert => "puppet:///apache/ssl-certs/chain-TCS.pem",
}

apache::vhost-ssl { « site2.bordeaux.inria.fr":
                ensure => present,
                ip_address => « X.X.X.X",
                cert => "puppet:///apache/ssl-certs/site2.inria.fr.pem",
                certkey => "puppet:///apache/ssl-certs/site2.inria.fr.key",
                cacert => "puppet:///apache/ssl-certs/chain-TCS.pem
}
```

Templates

- Fichier à contenu dynamique
- Contient du code Ruby (.erb)

```
"Debian": {
    $nrpedir     = "/etc/nagios/"
  }
  "CentOS": {
    $nrpedir     = "/etc/nagios/"
  }
```

Nrpe.cfg.erb

```
include=<%= nrpedir %>nrpe_command.cfg
```

Bonnes pratiques

- Gestion de versions
- Surveillez votre infrastructure
- Passez de webrick à Mongrel ou Passenger
- Documentez !

Bonnes pratiques: versionning

- Utilisez des outils tels que Subversion,Git, etc.

- Utilisez des branches :
  - Production
  - Test

```
[testing]

manifest = /var/lib/puppet/testing/manifests/site.pp
modulepath = /var/lib/puppet/testing/modules/
pluginpath = /var/lib/puppet/testing/plugins/

[production]

manifest = /var/lib/puppet/production/manifests/site.pp
modulepath = /var/lib/puppet/production/modules/
pluginpath = /var/lib/puppet/production/plugins
```

Bonnes pratiques: surveillance

- Les logs
- Nagios, RRD
- Les mails
- PuppetDashboard

Bonnes pratiques: surveillance

- Les logs

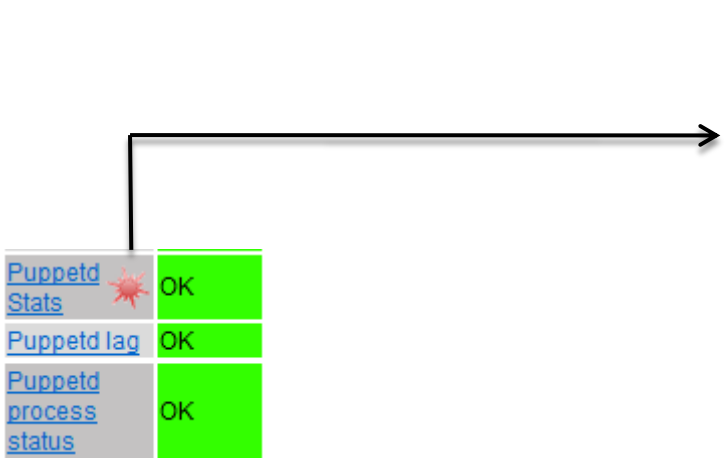
```
debug: Finishing transaction 23474806028980 with 0 changes
debug: Storing state
debug: Stored state in 0.12 seconds
warning: Value of 'preferred_serialization_format' (pson) is invalid for
report, using default (marshal)
debug: report supports formats: b64_zlib_yaml marshal raw yaml; using
marshal
notice: Finished catalog run in 4.52 seconds
```

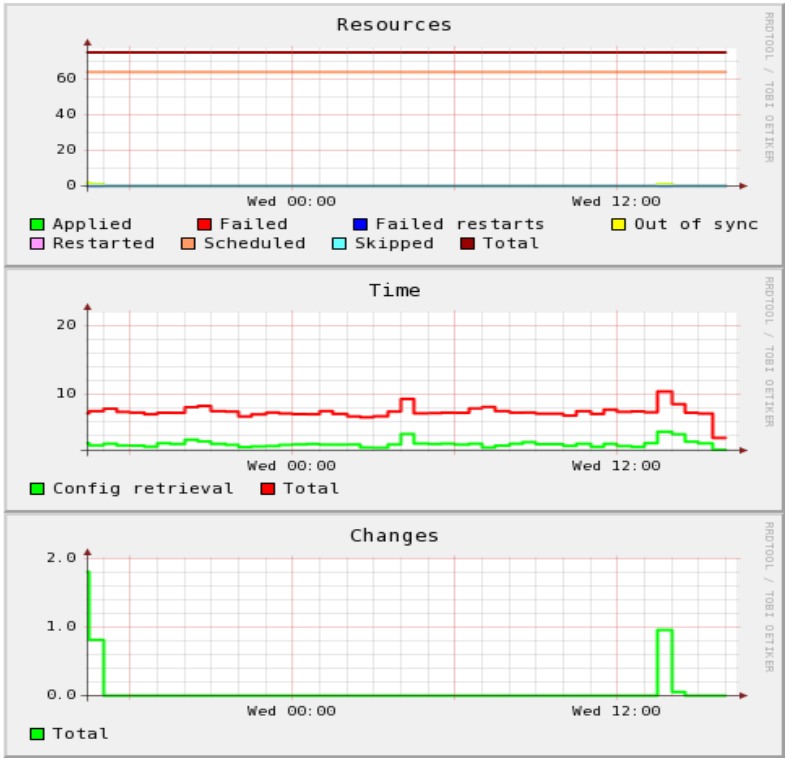
Bonnes pratiques: surveillance

- Les logs
- Nagios, RRD
- Les mails
- PuppetDashboard

Bonnes pratiques: surveillance

- Nagios, RRD





Bonnes pratiques: surveillance

- Les logs
- Nagios, RRD
- Les mails
- PuppetDashboard

Bonnes pratiques: surveillance

- Les mails

From: root@puppet.bordeaux.inria.fr
Subject: **Puppet Report for pinocchio.bordeaux.inria.fr**
To: root@puppet2.bordeaux.inria.fr
Date: Mon, 29 Mar 2010 16:16:16 +0200 (CEST)

Mon Mar 29 16:16:16 +0200 2010
//ssh::server/Sshd_config[puppet]/File[/etc/ssh/sshd_config]
(err): Failed to retrieve current state of resource:
Could not retrieve information from source(s)
puppet:///ssh/sshd_config.pinocchio, puppet:///ssh/sshd_config at
/var/lib/puppet/production/modules/ssh/manifests/init.pp:58

Bonnes pratiques: surveillance

- Les logs
- Nagios, RRD
- Les mails
- PuppetDashboard

Bilan

- Déployé à l'INRIA Bordeaux depuis Avril 2009
- Une refonte en cours
- 55 machines (serveurs & postes de travail) gérées
  - Centos, Debian, Ubuntu
  - Physique, Xen, OpenVZ
- Services
    - Dépots (yum, apt-get), ssh, ntp, postfix, logwatch, apache, php, nagios, autofs, nut et puppet

Liens

- Site PuppetLabs http://www.puppetlabs.com
- Site Github http://github.com
- Livre: Pulling String with Puppet http://apress.com/book/view/1590599780