

Puppet : Solution de gestion de fichier de configuration

De Deimos.fr / Bloc Notes Informatique.

Version du 1 juillet 2011 à 21:00 par Deimos (discuter | contributions | bloquer)

(diff) ← Version précédente | Voir la version courante (diff) | Version suivante → (diff)

0

J'aime

Sommaire

- 1 Introduction
- 2 Hiérarchie de Puppet
- 3 Installation
 - 3.1 Serveur
 - 3.2 Fonctionnement de puppet avec Mongrel et NGINX
 - 3.2.1 Installation
 - 3.2.2 Configuration
 - 3.3 Clients
 - 3.3.1 Debian
 - 3.3.2 Solaris
- 4 Configuration
 - 4.1 Serveur
 - 4.1.1 site.pp
 - 4.2 Client
 - 4.2.1 puppet.conf
 - 4.2.1.1 Debian
 - 4.2.1.2 Solaris
 - 4.2.2 Certificats
 - 4.2.2.1 Création d'un certificat
 - 4.2.2.2 Ajout d'un client puppet au serveur
 - 4.2.2.3 Révoquer un certificat
- 5 Surveillance des processus
 - 5.1 Détermination de l'état des noeuds
- 6 Création d'un noeud
- 7 Les modules
 - 7.1 Initialisation d'un module
 - 7.2 sudo
 - 7.2.1 init.pp
 - 7.2.1.1 Amélioration d'un module
 - 7.2.2 files
 - 7.3 ssh
 - 7.3.1 init.pp
 - 7.3.2 templates
 - 7.4 Nsswitch
 - 7.4.1 factor
 - 7.4.2 manifests
 - 7.4.3 templates

- 7.5 Nagios NRPE + plugins
 - 7.5.1 init.pp
 - 7.5.2 files
 - 7.5.3 templates
- 7.6 Munin
 - 7.6.1 Munin Interfaces
- 7.7 Importation d'un module
 - 7.7.1 Importer tous les modules d'un coup
- 8 Configuration avancée
 - 8.1 Création de Facts
 - 8.2 Outrepasser des restrictions
 - 8.3 Puppet Push
- 9 FAQ
 - 9.1 err: Could not retrieve catalog from remote server: hostname was not match with the server certificate
- 10 Ressources

1 Introduction



Puppet est une application très pratique... C'est ce que l'on pourrait retrouver dans les entreprises avec de grands volumes de serveurs, où le système d'information est « industrialisé ».

Puppet permet d'automatiser un grand nombre de tâche d'administration, comme l'installation de logiciels, de services ou encore de modifier des fichiers.

Puppet permet de faire cela de manière centralisée ce qui permet d'administrer et de mieux contrôler un grand nombre de serveur hétérogènes ou homogènes.

Puppet fonctionne en mode Client / Serveur.

Sur chaque machine un client va être installé et c'est lui qui va contacter le PuppetMaster, le serveur, par le biais de communication HTTPS, et donc SSL, un système pki est fourni.

Puppet a été développé en Ruby le rendant multiplateforme : bsd (free, macos ...) ,linux (redhat, debian, suse ...), sun (opensolaris ...)

Reductive Labs (<http://reductivelabs.com/>) , la société éditant Puppet, a développé un produit complémentaire, nommé Factor.

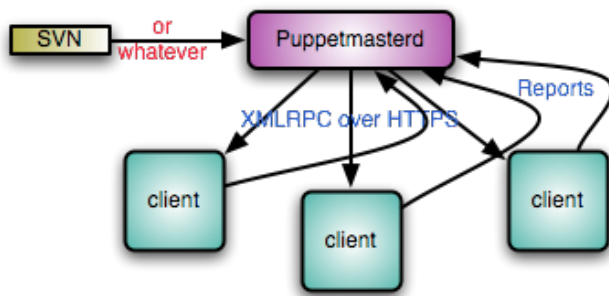
Cette application permet de lister des éléments propres aux systèmes administrés, comme le nom de machine, l'adresse ip, la distribution, des variables d'environnement utilisables dans les templates de puppet.

Puppet gérant des templates, on peut rapidement comprendre l'utilité de Factor, si par exemple on gère une ferme de serveurs de messagerie, qui nécessite un paramétrage contenant par exemple le nom de la machine.

Là un template combiné avec des variables d'environnements s'avère tout à fait utile.

Enfin bref Puppet combiné à Factor me semble une solution très intéressante pour simplifier l'administration de systèmes.

Voici un schéma de fonctionnement de puppet :



Pour la configuration de Puppet, si vous souhaitez utiliser un IDE, il existe Geppetto (<http://www.puppetlabs.com/blog/geppetto-a-puppet-ide/>) .

2 Hiérarchie de Puppet

Avant d'aller plus loin, j'ai pompé depuis le site officiel le fonctionnement de l'arborescence de puppet :

All Puppet data files (modules, manifests, distributable files, etc) should be maintained in a Subversion or CVS repository (or your favorite Version Control System). The following hierarchy describes the layout one should use to arrange the files in a maintainable fashion:

- `/manifests/`: this directory contains files and subdirectories that determine the manifest of individual systems but do not logically belong to any particular module. Generally, this directory is fairly thin and alternatives such as the use of LDAP or other external node tools can make the directory even thinner. This directory contains the following special files:
 - `site.pp`: first file that the Puppet Master parses when determining a server's catalog. It imports all the underlying subdirectories and the other special files in this directory. It also defines any global defaults, such as package managers. See sample `site.pp`.
 - `templates.pp`: defines all template classes. See also `terminology:template classes`. See sample `templates.pp`.
 - `nodes.pp`: defines all the nodes if not using an external node tool. See sample `nodes.pp`.
- `/modules/{modulename}/`: houses puppet modules in subdirectories with names matching that of the module name. This area defines the general building blocks of a server and contains modules such as for `openssh`, which will generally define classes `openssh::client` and `openssh::server` to setup the client and server respectively. The individual module directories contains subdirectories for manifests, distributable files, and templates. See `modules organization`, `terminology:module`.
- `/modules/user/`: A special module that contains manifests for users. This module contains a special subclass called `user::virtual` which declares all the users that might be on a given system in a virtual way. The other subclasses in the user module are classes for logical groupings, such as `user::unixadmins`, which will realize the individual users to be included in that group. See also `naming conventions`, `terminology:realize`.
- `/services/`: this is an additional modules area that is specified in the module path for the puppetmaster. However, instead of generic modules for individual services and bits of a server, this module area is used to model servers specific to enterprise level infrastructure services (core infrastructure services that your IT department provides, such as `www`, enterprise directory, file server, etc). Generally, these classes will include the modules out of `/modules/` needed as part of the catalog (such as `openssh::server`, `postfix`, `user::unixadmins`, etc). The files section for these modules is used to distribute configuration files specific to the enterprise infrastructure service such as `openldap` schema files if the module were for the enterprise directory. To avoid namespace collision with the general modules, it is recommended that these modules/classes are prefixed with `s_` (e.g. `s_ldap` for the enterprise directory server module)
- `/clients/`: similar to the `/services/` module area, this area is used for modules related to modeling servers for external clients (departments outside your IT department). To avoid namespace collision, it is recommended that these modules/classes are prefixed with `c_`.


- /notes/: this directory contains notes for reference by local administrators.
- /plugins/: contains custom types programmed in Ruby. See also terminology:plugin-type.
- /tools/: contains scripts useful to the maintenance of Puppet.

3 Installation

3.1 Serveur


La version utilisée du master doit être la même que celle des postes clients. Il est très fortement recommandé d'utiliser une version supérieure ou égale à 0.25.4 (correctif de nombreux problèmes de performance). Pour cela, sur Debian, il faudra installer la version disponible en backport, et la bloquer pour éviter qu'une upgrade malencontreuse ne change sa version (utilisation des "pin locks").

Pour le moment, il faut configurer le fichier /etc/hosts avec l'ip du serveur :

 /etc/hosts

```
...
192.168.0.93 puppet.deimos.fr puppet
...
```


Ajouter les informations suivantes (/etc/apt/preferences) pour utiliser le backport par défaut sur puppet et puppetmaster:

 /etc/apt/preferences

```
0. Package: puppet*
1. Pin: release v=0.25.4-2~bpo50+1,a=lenny-backports
2. Pin-Priority: 999
```


Note : Vérifiez que l'horloge du puppetmaster (et les client aussi bien sûr) est bien à jour/synchronisée. Il peut y avoir un problème avec les certificats qui seront non reconnus/acceptés si il y a un décalage (faire un `dpkg-reconfigure tz-data`).

Installer puppetmaster :




```
echo "deb http://www.backports.org/debian lenny-backports main contrib non-free" >> /etc/apt/sources.list
aptitude update
aptitude install debian-backports-keyring
aptitude update
aptitude install puppetmaster mongrel
```

Attention, il y a une dépendance (non nécessaire, mais forcée par le maintenir du package debian) sur puppet. Puppet sera donc aussi installé. Il sera utile de désactiver le daemon en modifiant le fichier /etc/default/puppet et en mettant START à "no" :

 /etc/default/puppet

```
# Defaults for puppet - sourced by /etc/init.d/puppet
# Start puppet on boot?
START=no
# Startup options
DAEMON_OPTS="-w 5"
```

On peut vérifier que puppetmaster est bien installé en lançant 'factor' ou la présence des fichiers SSL (dans /var/lib/puppet) ou le lancer directement et regarder le résultat dans /var/log/daemon.log ou avec la commande suivante :

 puppetmasterd

```
> puppetmasterd --verbose --no-daemonize
info: Starting server for Puppet version 0.24.5
info: mount[files]: allowing 192.168.0.* access
info: mount[files]: allowing 10.101.0.* access
info: Listening on port 8140
notice: Starting Puppet server version 0.24.5
```

3.2 Fonctionnement de puppet avec Mongrel et NGINX

3.2.1 Installation

Il faut installer nginx. Pour cela nous allons encore une fois utiliser la version des backports :

 /etc/apt/preferences

```
...
Package: nginx
Pin: release v=0.7.65-2-bpo50+1,a=lenny-backports
Pin-Priority: 999
```

Puis nous allons l'installer

 aptitude

```
aptitude update
aptitude install nginx
```

3.2.2 Configuration

Modification du fichier /etc/default/puppetmaster :

```

/etc/default/puppetmaster

# Defaults for puppetmaster - sourced by /etc/init.d/puppet
# Start puppet on boot?
START=yes
# Startup options
DAEMON_OPTS=""
# What server type to run
# Options:
#     webrick (default, cannot handle more than ~30 nodes)
#     mongrel (scales better than webrick because you can run
#             multiple processes if you are getting
#             connection-reset or End-of-file errors, switch to
#             mongrel. Requires front-end web-proxy such as
#             apache, nginx, or pound)
# See: http://reductivelabs.com/trac/puppet/wiki/UsingMongrel
SERVERTYPE=mongrel
# How many puppetmaster instances to start? Its pointless to set this
# higher than 1 if you are not using mongrel.
PUPPETMASTERS=4
# What port should the puppetmaster listen on (default: 8140). If
# PUPPETMASTERS is set to a number greater than 1, then the port for
# the first puppetmaster will be set to the port listed below, and
# further instances will be incremented by one
#
# NOTE: if you are using mongrel, then you will need to have a
# front-end web-proxy (such as apache, nginx, pound) that takes
# incoming requests on the port your clients are connecting to
# (default is: 8140), and then passes them off to the mongrel
# processes. In this case it is recommended to run your web-proxy on
# port 8140 and change the below number to something else, such as
# 18140.
PORT=18140

```

Après (re-)lancement, on doit pouvoir voir les sockets attachées :

```

netstat

> netstat -pvltpn
Connexions Internet actives (seulement serveurs)
Proto Recv-Q Send-Q Adresse locale Adresse distante Etat PID/Program name
tcp 0 0 0.0.0.0:41736 0.0.0.0:* LISTEN 2029/rpc.statd
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 2018/portmap
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 2333/sshd
tcp 0 0 127.0.0.1:18140 0.0.0.0:* LISTEN 10059/ruby
tcp 0 0 127.0.0.1:18141 0.0.0.0:* LISTEN 10082/ruby
tcp 0 0 127.0.0.1:18142 0.0.0.0:* LISTEN 10104/ruby
tcp 0 0 127.0.0.1:18143 0.0.0.0:* LISTEN 10126/ruby
tcp6 0 0 :::22 :::* LISTEN 2333/sshd

```

Ajout de la ligne suivante dans le fichier /etc/puppet/puppet.conf :

```

/etc/puppet/puppet.conf

[main]
logdir=/var/log/puppet
vardir=/var/lib/puppet
ssldir=/var/lib/puppet/ssl
rundir=/var/run/puppet
factpath=$vardir/lib/facter
pluginsync=false

[puppetmasterd]
templatedir=/var/lib/puppet/templates

```

```
ssl_client_header=HTTP_X_SSL_SUBJECT
autosign=false
```

Modifier/ajouter la configuration suivante dans /etc/nginx.conf :



/etc/nginx/nginx.conf

```
0. user daemon daemon;
1. worker_processes 4;
2.
3. error_log /var/log/nginx/puppet.log notice;
4. pid /var/run/nginx.pid;
5.
6. events {
7.     worker_connections 1024;
8. }
9.
10.
11. http {
12.     # include /etc/mime.types;
13.     default_type application/octet-stream;
14.
15.     # no sendfile on OSX uncomment
16.     #this if your on linux or bsd
17.     sendfile on;
18.     tcp_nopush on;
19.
20.     # Look at TLB size in /proc/cpuinfo (Linux) for the 4k pagesize
21.     large_client_header_buffers 16 4k;
22.     proxy_buffers 128 4k;
23.
24.     # if you adjust this setting to something higher
25.     # you should as well update the proxy_read_timeout
26.     # in the server config part (see below)
27.     # Otherwise nginx will rerequest a manifest compile.
28.     keepalive_timeout 65;
29.     tcp_nodelay on;
30.
31.     upstream puppet-production {
32.         server 127.0.0.1:18140;
33.         server 127.0.0.1:18141;
34.         server 127.0.0.1:18142;
35.         server 127.0.0.1:18143;
36.     }
37.
38.
39.     server {
40.         listen 8140;
41.
42.         ssl on;
43.         ssl_certificate /var/lib/puppet/ssl/certs/puppet.deimos.fr.pem;
44.         ssl_certificate_key /var/lib/puppet/ssl/private_keys/puppet.deimos.fr.p
45.         ssl_client_certificate /var/lib/puppet/ssl/ca/ca.crt.pem;
46.         ssl_ciphers SSLv2:-LOW:-EXPORT:RC4+RSA;
47.         ssl_session_cache shared:SSL:8m;
48.         ssl_session_timeout 5m;
49.
50.         ssl_verify_client optional;
51.
```

```

52.      # obey to the Puppet CRL
53.      ssl_crl          /var/lib/puppet/ssl/ca/ca_crl.pem;
54.
55.      root             /var/empty;
56.      access_log       /var/log/nginx/access-8140.log;
57.      #rewrite_log     /var/log/nginx/rewrite-8140.log;
58.
59.      # Variables
60.      # $ssl_cipher returns the line of those utilized it is cipher for establish
61.      # $ssl_client_serial returns the series number of client certificate for es
62.      # $ssl_client_s_dn returns line subject DN of client certificate for establ
63.      # $ssl_client_i_dn returns line issuer DN of client certificate for establi
64.      # $ssl_protocol returns the protocol of established SSL-connection
65.
66.      location / {
67.          proxy_pass      http://puppet-production;
68.          proxy_redirect  off;
69.          proxy_set_header Host                $host;
70.          proxy_set_header X-Real-IP          $remote_addr;
71.          proxy_set_header X-Forwarded-For    $proxy_add_x_forwarded_for;
72.          proxy_set_header X-Client-Verify    SUCCESS;
73.          proxy_set_header X-SSL-Subject      $ssl_client_s_dn;
74.          proxy_set_header X-SSL-Issuer       $ssl_client_i_dn;
75.          proxy_read_timeout 65;
76.      }
77.  }
78. }

```

Pour vérifier que les daemons tournent correctement, vous devez avoir les sockets suivantes ouvertes :

 netstat

```

0. > netstat -vlptn
1. Connexions Internet actives (seulement serveurs)
2. Proto Recv-Q Send-Q Adresse locale      Adresse distante     Etat      PID
3. tcp      0      0 0.0.0.0:41736        0.0.0.0:*             LISTEN    202
4. tcp      0      0 0.0.0.0:8140        0.0.0.0:*             LISTEN    102
5. tcp      0      0 0.0.0.0:8141        0.0.0.0:*             LISTEN    102
6. tcp      0      0 0.0.0.0:111         0.0.0.0:*             LISTEN    201
7. tcp      0      0 0.0.0.0:22          0.0.0.0:*             LISTEN    233
8. tcp      0      0 127.0.0.1:18140     0.0.0.0:*             LISTEN    100
9. tcp      0      0 127.0.0.1:18141     0.0.0.0:*             LISTEN    100
10. tcp     0      0 127.0.0.1:18142     0.0.0.0:*             LISTEN    101
11. tcp     0      0 127.0.0.1:18143     0.0.0.0:*             LISTEN    101
12. tcp6    0      0 :::22               :::*                  LISTEN    233

```

3.3 Clients

Pour les clients, c'est simple aussi. Mais avant ajoutez la ligne du serveur dans le fichier hosts :



```
...
192.168.0.93 puppet.deimos.fr puppet
```

3.3.1 Debian

Vérifier que le fichier /etc/hosts contient bien le hostname de la machine cliente :



```
apt-get install puppet
```

3.3.2 Solaris

Le client Puppet dans la version stable de blastwave est trop ancien (0.23). Il faudra donc installer Puppet (et Factor) par l'intermédiaire du gestionnaire standard de Ruby: gem.

Pour cela, il faudra au préalable installer ruby avec la commande suivante:



```
pkg-get -i ruby
```

Attention, vérifier que rubygems n'est pas déjà installé, sinon le supprimer :



```
pkg-get -r rubygems
```

puis installer une version plus à jour à partir des sources:



```
wget http://rubyforge.org/frs/download.php/45905/rubygems-1.3.1.tgz
gzcat rubygems-1.3.1.tgz | tar -xf -
cd rubygems-1.3.1
ruby setup.rb
gem --version
```

Installer puppet avec la commande et l'argument -p si vous avez un proxy :



gem

```
gem install puppet --version '0.25.4' -p http://proxy:3128/
```

Il faut modifier/ajouter quelques commandes qui ne sont pas par défaut sur Solaris, pour que Puppet fonctionne mieux:

- Créer un lien pour uname et puppetd :



ln

```
ln -s /usr/bin/uname /usr/bin/  
ln -s /opt/csw/bin/puppetd /usr/bin/
```

- Créer un script appelé /usr/bin/dnsdomainname :



/usr/bin/dnsdomainname

```
0. #!/usr/bin/bash  
1. DOMAIN="`/usr/bin/domainname 2> /dev/null`"  
2. if [ ! -z "$DOMAIN" ]; then  
3.     echo $DOMAIN | sed 's/^[^.]*/./'  
4. fi
```



chmod

```
chmod 755 /usr/bin/dnsdomainname
```

Ensuite, la procédure est la même que pour les autres OS, c'est à dire, modifier /etc/hosts pour inclure puppet.deimos.fr, et lancer:



puppetd

```
puppetd --verbose --no-daemon --test --server puppet.deimos.fr
```

A ce stade là, si ça ne fonctionne pas, c'est tout simplement qu'il faut modifier la configuration (puppet.conf) de votre client.

4 Configuration

4.1 Serveur

Donnez les autorisations des machines clientes dans le fichier `/etc/puppet/fileserver.conf` :



`/etc/puppet/fileserver.conf`

```
## This file consists of arbitrarily named sections/modules
## defining where files are served from and to whom

## Define a section 'files'
## Adapt the allow/deny settings to your needs. Order
## for allow/deny does not matter, allow always takes precedence
## over deny
#[files]
  path /etc/puppet/files
  allow 192.168.0.*
  allow 10.101.0.*
  allow *.example.com
  deny *.evil.example.com
  allow 192.168.0.0/24

#[plugins]
  allow *.example.com
  deny *.evil.example.com
  allow 192.168.0.0/24
```

Créons les fichiers manquants :



```
touch /etc/puppet/manifests/{common.pp,modules.pp,site.pp}
```

Puis on va les renseigner un par un. On ajoute dans le `common.pp` ce qu'il faut pour qu'il ignore les parties faisant appel à des repository :



`/etc/puppet/manifests/common.pp`

```
File { ignore => ['.svn', '.git', 'CVS' ] }
```


Ensuite je vais définir ici mes modules de base :



`/etc/puppet/manifests/modules.pp`

```
import "common"
```

On demande de charger tous les modules présents dans le dossier `modules` :


 /etc/puppet/manifests/site.pp

```
import "modules.pp"
```

4.1.1 site.pp

Ce qui suit est optionnel du fait que la conf a déjà été faites plus haut.

Maintenant, nous allons manipuler un fichier. Il est quasiment identique à la conf de base :

 /etc/puppet/manifests/site.pp

```
0. # /etc/puppet/manifests/site.pp
1.
2. import "modules"
3. import "nodes"
4.
5. # The filebucket option allows for file backups to the server
6. filebucket { main: server => 'puppet.deimos.fr' }
7.
8. # Set global defaults - including backing up all files to the main filebucket and a
9. File { backup => main }
10. Exec { path => "/usr/bin:/usr/sbin:/bin:/sbin" }
```

Ici on vient lui dire d'importer les confs modules et nodes, le filebucket et les globales par défaut.

Il faut savoir que toute cette configuration est propre au serveur puppet, donc global. Tous ce que nous mettrons dedans pourra être hérité.

On relance le puppetmaster pour être sûr que les modifications côté serveur ont bien été prises en compte.


4.2 Client

Chaque client doit avoir son entrée dans le serveur DNS (tout comme le serveur) !

4.2.1 puppet.conf

4.2.1.1 Debian


Le fichier de configuration doit contenir l'adresse du serveur :

 /etc/puppet/puppet.conf

```
...
[puppet]
server = puppet.deimos.fr
```

4.2.1.2 Solaris

Pour la partie Solaris, il a fallu pas mal adapter la configuration :

 /etc/puppet/puppet.conf

```
0. [main]
1.   logdir=/var/log/puppet
2.   vardir=/var/opt/csw/puppet
3.   rundir=/var/run/puppet
4.   # sslidir=/var/lib/puppet/ssl
5.   sslidir=/etc/puppet/ssl
6.   # Where 3rd party plugins and modules are installed
7.   libdir = $vardir/lib
8.   templatedir=$vardir/templates
9.   # Turn plug-in synchronization on.
10.  pluginsync = true
11.  pluginsource = puppet://$server/plugins
12.  factpath = /var/puppet/lib/facter
13.
14. [puppetd]
15.  report=true
16.  server=puppet.deimos.fr
17.  # certname=puppet.deimos.fr
18.  # enable the marshal config format
19.  config_format=marshal
20.  # different run-interval, default= 30min
21.  # e.g. run puppetd every 4 hours = 14400
22.  runinterval = 14400
23.  logdest=/var/log/puppet/puppet.log
```

4.2.2 Certificats

Puppet travaille avec des certificats pour les échanges clients/serveurs. Il va donc falloir générer des clefs SSL sur le serveur et faire un échange ensuite avec tous les clients. L'utilisation des clefs SSL nécessite donc une bonne configuration de votre serveur DNS. Vérifiez donc bien à ce que :

- Le nom du serveur soit définitif
- Le nom du serveur soit accessible via puppet.mydomain.com (je continuerais la configuration pour moi avec puppet.deimos.fr)

4.2.2.1 Création d'un certificat

- Nous allons donc créer notre certificat (**normalement déjà fait lorsque l'installation est faite sur Debian**) :

 puppetca

```
puppetca -g puppet.deimos.fr
```

- Dans le cas où vous souhaitez valider un certificat avec plusieurs noms de domaine, il va falloir procéder comme ceci :

```
puppetca
```

```
puppetca -g --certdnsnames puppet:scar.deimos.fr puppet.deimos.fr
```

Insérez les noms dont vous avez besoin les uns à la suite des autres. Je vous rappelle que par défaut, les clients vont chercher **puppet.mydomain.com**, donc n'hésitez pas à rajouter des noms si besoin.

Vous pouvez ensuite vérifier que le certificat contient bien les 2 noms :

```
openssl
```

```
> openssl x509 -text -in /var/lib/puppet/ssl/certs/puppet.deimos.fr.pem | grep DNS
DNS:puppet, DNS:scar.deimos.fr, DNS:puppet.deimos.fr
```

Vous pouvez voir d'éventuelles erreurs de certificats en vous connectant :

```
openssl
```

```
openssl s_client -host puppet -port 8140 -cert /path/to/ssl/certs/node.domain.com.pem -key /path/to/ssl/private_keys/r
```

Note: N'oubliez pas de redémarrer votre serveur web si vous faites le changement de certificats

4.2.2.2 Ajout d'un client puppet au serveur

Pour le certificat, c'est simple, nous allons faire une demande de certificats :

```
puppetd
```

```
is$ puppetd --server puppet.deimos.fr --waitforcert 60 --test
notice: Ignoring cache
info: Caching catalog at /var/lib/puppet/state/localconfig.yaml
notice: Starting catalog run
notice: //information_class/Exec[echo running on debian-puppet.deimos.fr is a Debian with ip 192.168.0.106. Message is
notice: Finished catalog run in 0.45 seconds
```

Maintenant nous allons nous connecter **sur le serveur** et lancer cette commande :

 puppetca

```
$ puppetca -l  
debian-puppet.deimos.fr
```

Je vois ici par exemple que j'ai un host qui veut faire un échange de clefs afin ensuite d'obtenir les configurations qui lui sont dues. Seulement il va falloir l'autoriser. Pour ce faire :



```
puppetca -s debian-puppet.deimos.fr
```

La machine est donc maintenant acceptée et peut aller chercher des confs sur le serveur Puppet.

Si on veut refuser un noeud qui est en attente :



```
puppetca -c debian-puppet.deimos.fr
```

Si on veut voir la liste de tous les noeuds autorisés, puppetmaster les tient enregistrés ici :



```
/var/lib/puppet/ssl/ca/inventory.txt
```

```
0. 0x0001 2010-03-08T15:45:48GMT 2015-03-07T15:45:48GMT /CN<nowiki>=</nowiki>puppet.de.  
1. 0x0002 2010-03-08T16:36:16GMT 2015-03-07T16:36:16GMT /CN<nowiki>=</nowiki>node1.dei  
2. 0x0003 2010-03-08T16:36:25GMT 2015-03-07T16:36:25GMT /CN<nowiki>=</nowiki>node2.dei  
3. 0x0004 2010-04-14T12:41:24GMT 2015-04-13T12:41:24GMT /CN<nowiki>=</nowiki>node3.dei
```

4.2.2.3 Révoquer un certificat

Si vous souhaitez révoquer un certificat, voici comment faire :



```
puppetca
```

```
puppetca -r ma_machine  
puppetca -c ma_machine
```

Simple non ? :-)

Si vous souhaitez réassigner de nouveau, supprimez côté client le dossier ssl dans /etc/puppet. Ensuite vous pouvez relancer une demande de certificat.

5 Surveillance des processus

Sur les clients, il n'est pas nécessaire d'utiliser un logiciel de surveillance, puisque les daemons puppetd ne sont lancés qu'à la main ou par tâche CRON. Sur le serveur, il est important que le processus puppetmaster soit toujours présent. On pourra utiliser le logiciel 'monit', qui pourra redémarrer automatiquement le processus en cas de problème.


5.1 Détermination de l'état des noeuds

Pour voir s'il y a des problèmes sur un noeud, on pourra lancer manuellement la commande suivante :

 puppetd

```
puppetd --no-daemon --verbose --onetime
```

Si l'on souhaite connaître le dernier état/résultat d'une mise à jour puppet, on pourra utiliser le système de 'report' une fois activé (sur les clients) :

 /etc/puppet/puppet.conf

```
...
report = true
...
```

En activant le reporting, à chaque exécution du daemon puppetd, un compte rendu sera envoyé sur le puppetmaster dans un fichier au format YAML, dans le répertoire /var/lib/puppet/reports/NOM_MACHINE.

Voici un exemple de fichier de report, facilement transformable en dictionnaire avec le module yaml de python (ou ruby) :

 /var/lib/puppet/reports/deb-puppet-client.deimos.fr

```
0. --- !ruby/object:Puppet::Transaction::Report
1. host: deb-puppet-client.deimos.fr
2. logs:
3. - !ruby/object:Puppet::Util::Log
4.   level: :info
5.   message: Applying configuration version '1275982371'
6.   source: Puppet
7.   tags:
8.   - info
9.   time: 2010-06-08 11:37:59.521132 +02:00
```



```
10. - !ruby/object:Puppet::Util::Log
11. file: /etc/puppet/modules/sudo/manifests/init.pp
12. level: :err
13. line: 11
14. message: "Failed to retrieve current state of resource: Could not retrieve inform
15. source: //sudo/File[/etc/sudoers]
```

6 Création d'un noeud

Maintenant nous allons autoriser les noeuds sur le serveur :



/etc/puppet/manifests/nodes.pp

```
0. # /etc/puppet/manifests/nodes.pp
1.
2. node basenode {
3.   include sudo
4. }
5.
6. # One node configuration
7. node 'debian-puppet.deimos.fr' inherits basenode {
8. }
9.
10. #node 'node1.deimos.fr', 'node2.deimos.fr' inherits basenode {
11. #   $ntp_localisation = 'us'
12. #   include general
13. #}
14.
15. # One domaine with x nodes configuration using regex
16. #node /\.*\.deimos\.fr$/ inherits basenode {
17. #}
```

Nous avons donc ici créer 2 noeuds. 1 qui n'en est pas un, c'est à dire basenode, qui va plutôt nous servir à inclure tous les services que nous voudrions déployer sur tous les hosts et un vrai noeud (debian-puppet.deimos.fr) qui lui héritera de basenode.

Si nous souhaitons aller plus loin, on peut également utiliser les regex (voir la partie de la configuration commentée), c'est également possible :-). Dès lors, des combinaisons complexes peuvent être faite.

Pour info, la configuration peut être intégrée dans LDAP (<http://reductivelabs.com/trac/puppet/wiki/LDAPNodes>) .

7 Les modules

Il est recommandé de créer des modules pour chaque service afin de rendre la configuration plus souple. Ca fait partie de certaines best practices.

Je vais aborder ici différentes techniques en essayant de garder un ordre croissant de difficulté.

7.1 Initialisation d'un module

Nous allons donc créer sur le serveur, l'arborescence adéquate. Pour cet exemple, nous allons partir avec sudo, mais vous pouvez choisir autre chose si vous souhaitez :



```
mkdir -p /etc/puppet/modules/sudo/{manifests,files}
```

N'oubliez pas que ceci est nécessaire pour chaque module.

7.2 sudo

7.2.1 init.pp

Le fichier init.pp est le cœur de notre module, renseignez le comme ceci pour le moment :



/etc/puppet/modules/sudo/manifests/init.pp

```
0. # sudo.pp
1. class sudo {
2.     # OS detection
3.     $sudoers_file = $operatingsystem ? {
4.         Solaris => "/opt/csw/etc/sudoers",
5.         default => "/etc/sudoers"
6.     }
7.
8.     # Sudoers file declaration
9.     file { "$sudoers_file":
10.         owner    => root,
11.         group    => root,
12.         mode     => 640,
13.         source   => $operatingsystem ? {
14.             Solaris => "puppet:///modules/sudo/sudoers.solaris",
15.             default => "puppet:///modules/sudo/sudoers"
16.         }
17.     }
18.
19.     # Symlink for solaris
20.     case $operatingsystem {
21.         Solaris: {
22.             file { "/opt/sfw/bin/sudo":
23.                 ensure => "/opt/csw/bin/sudo"
24.             }
25.         }
26.     }
27. }
```

Je vais essayer d'être clair dans les explications :

- Nous déclarons une classe sudo
- On fait une détection d'OS. Cette déclaration est hérité d'une variable (\$sudoers_file) qui nous permet de déclarer différents path du fichier sudoers.
- Qui comprends un fichier de configuration se trouvant dans (name) /opt/csw/etc/sudoers sur le système cible (pour cette conf, c'est sur Solaris, a vous d'adapter)
- Le fichier doit appartenir au user et groupe root avec les droits 440.
- La source de ce fichier (que nous n'avons pas encore déposé) est disponible (source) à l'adresse puppet:///modules/sudo/sudoers (soit /etc/puppet/modules/sudo/files/sudoers). Il indique tout simplement l'emplacement des fichiers dans votre arborescence puppet qui utilise un mécanisme de système de fichier interne à Puppet.

7.2.1.1 Amélioration d'un module

Editons le fichier pour lui rajouter quelques demandes



```

0. # sudo.pp
1. class sudo {
2.
3.     # Check if sudo is the latest version
4.     package { sudo: ensure => latest }
5.
6.     # OS detection
7.     $sudoers_file = $operatingsystem ? {
8.         Solaris => "/opt/csw/etc/sudoers",
9.         default => "/etc/sudoers"
10.    }
11.
12.    # Sudoers file declaration
13.    file { "$sudoers_file":
14.        owner    => root,
15.        group    => root,
16.        mode     => 640,
17.        source   => $operatingsystem ? {
18.            Solaris => "puppet:///modules/sudo/sudoers.solaris",
19.            default => "puppet:///modules/sudo/sudoers"
20.        },
21.        require  => Package["sudo"]
22.    }
23. }

```

- require => Package["sudo"] : on lui demande d'installer le package sudo s'il n'est pas installé
- package { sudo: ensure => latest } : on lui demande de vérifier que c'est bien la dernière version

7.2.2 files

Maintenant, il faut ajouter les fichiers que nous souhaitons publier dans le dossier files (ici seulement sudoers) :



cp

```
cp /etc/sudoers /etc/puppet/modules/sudo/files/sudoers
cp /etc/sudoers /etc/puppet/modules/sudo/files/sudoers.solaris
```

Pour la faire simple, j'ai simplement copier le sudoers du serveur vers la destination qui correspond à ma conf décrite plus haut.

Si vous n'avez pas installé sudo sur le serveur, mettez un fichier sudoers qui vous convient dans /etc/puppet/modules/sudo/files/sudoers.

Dans les prochaines version, puppet supportera le protocole http et ftp pour aller chercher ces fichiers.

7.3 ssh

Vous devez initialiser le module avant de continuer.

7.3.1 init.pp

Ici je veux que ce soit mon fichier sshd_config qui m'intéresse :



/etc/puppet/modules/ssh/manifests/init.pp

```
0. #ssh.pp
1.
2. # SSH Class with all includes
3. class ssh {
4.     $ssh_default_port = ["22"]
5.     $ssh_allowed_users = "root"
6.     include ssh::config, ssh::key, ssh::service
7. }
8.
9. # SSHD Config file
10. class ssh::config {
11.     File {
12.         name => $operatingsystem ? {
13.             Solaris => "/etc/ssh/sshd_config",
14.             default => "/etc/ssh/sshd_config"
15.         },
16.     }
17.
18.     # Using templates for sshd_config
19.     file { sshd_config:
20.         content => $operatingsystem ? {
21.             default => template("ssh/sshd_config"),
22.             Solaris => template("ssh/sshd_config.solaris"),
23.         }
24.     }
25. }
26.
27. # SSH Key exchange
28. class ssh::key {
29.     $basedir = $operatingsystem ? {
30.         Solaris => "/.ssh",
```

```

31.     Debian => "/root/.ssh",
32.         Redhat => "/root/.ssh",
33.     }
34.
35.     # Make sur .ssh exist in root home dir
36.     file { "$basedir/":
37.         ensure => directory,
38.         mode   => 0700,
39.         owner  => root,
40.         group  => root,
41.         ignore => '.svn'
42.     }
43.
44.     # Check if authorized_keys key file exist or create empty one
45.     file { "$basedir/authorized_keys":
46.         ensure => present,
47.     }
48.
49.     # Check this line exist
50.     line { ssh_key:
51.         file => "$basedir/authorized_keys",
52.         line => "ssh-dss AAAAB3NzaC1....zG3ZA== root@puppet",
53.         ensure => present;
54.     }
55. }
56.
57. # Check service status
58. class ssh::service {
59.     service { ssh:
60.         name => $operatingsystem ? {
61.             Solaris => "svc:/network/ssh:default",
62.             default => ssh
63.         },
64.         ensure    => running,
65.         enable    => true
66.     }
67. }

```

Ensuite, par rapport à sudo, j'ai un notify qui permet d'automatiquement redémarrer le service lorsque le fichier est remplacé par une nouvelle version. C'est le service ssh avec l'option "ensure => running", qui va permettre la détection du changement de version et redémarrage.

7.3.2 templates

Du fait que nous utilisons des templates, nous allons avoir besoin de créer un dossier templates :

 mkdir

```
mkdir -p /etc/puppet/modules/ssh/templates
```

Ensuite nous allons créer 2 fichiers (sshd_config et sshd_config.solaris) car les configuration ne se comportent pas de la même manière (OpenSSH vs Sun SSH. Je n'aborderais cependant ici que la partie OpenSSH :



/etc/puppet/modules/ssh/templates

```

# Package generated configuration file
# See the sshd(8) manpage for details
#
# What ports, IPs and protocols we listen for
<% ssh_default_port.each do |val| -%>
!Port <%= val -%>
!<% end -%>
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
!Protocol 2
# HostKeys for protocol version 2
!HostKey /etc/ssh/ssh_host_rsa_key
!HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
!UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
!KeyRegenerationInterval 3600
!ServerKeyBits 768

# Logging
!SyslogFacility AUTH
!LogLevel INFO

# Authentication:
!LoginGraceTime 120
!PermitRootLogin yes
!StrictModes yes

!RSAAuthentication yes
!PubkeyAuthentication yes
#AuthorizedKeysFile          %h/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
!IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
!RhostsRSAAuthentication no
# similar for protocol version 2
!HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
!IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
!PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
!ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
!PasswordAuthentication yes

# Kerberos options
!KerberosAuthentication no
!KerberosGetAFSToken no
!KerberosOrLocalPasswd yes
!KerberosTicketCleanup yes

# GSSAPI options
!GSSAPIAuthentication no
!GSSAPICleanupCredentials yes

!X11Forwarding yes
!X11DisplayOffset 10
!PrintMotd no
!PrintLastLog yes
!TCPKeepAlive yes
!UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net

# Allow client to pass locale environment variables
!AcceptEnv LANG LC_*

!Subsystem sftp /usr/lib/openssh/sftp-server

!UsePAM yes
! # AllowUsers <%= ssh_allowed_users %>

```

Ici nous utilisons donc 2 types d'utilisation de templates. Une multi lignes a répétition est l'autre, plus simple avec un simple remplacement de variables :

- `ssh_default_port.each do` : permet de mettre une ligne de "Port num_port" a chaque port spécifié
- `ssh_allowed_users` : permet de donner une liste d'utilisateur

Ces méthodes s'avèrent simple et très efficaces. Petite subtilité :

- `-%>` : Lorsqu'une ligne se termine comme ceci, c'est qu'il ne va pas y avoir de saut de ligne grâce au - situé à la fin.
- `%>` : Il y aura un saut de ligne ici.

7.4 Nsswitch

Pour nsswitch, on va utiliser une technique avancée qui consiste à utiliser Factor (un built in qui permet de gagner beaucoup de temps). Factor propose d'avoir en gros des variables d'environnements spécifiques à Puppet qui permettent de faire des conditions suite à cela. Par exemple, je souhaite vérifier la présence d'un fichier qui va m'indiquer si mon serveur est en mode cluster (pour Sun Cluster) ou non et modifier le fichier nsswitch en fonction. Pour cela je vais donc utiliser factor.

7.4.1 factor

Nous allons créer la hiérarchie pour notre script factor :



mkdir

```
mkdir -p /etc/puppet/modules/nsswitch/lib/factor/
```

Puis nous allons créer notre fichier de fact :



/etc/puppet/modules/nsswitch/lib/factor/is_cluster.rb

```
0. # is_cluster.rb
1.
2. Factor.add("is_cluster") do
3.   setcode do
4.     #%x{/bin/uname -i}.chomp
5.     FileTest.exists?("/etc/cluster/nodeid")
6.   end
7. end
```

7.4.2 manifests

Ensuite nous allons spécifier l'utilisation d'un template :



/etc/puppet/modules/nsswitch/manifests/init.pp

```

0. class nsswitch {
1.     case $operatingsystem {
2.         Solaris: { include nsswitch::solaris }
3.         default: { }
4.     }
5. }
6.
7. class nsswitch::solaris {
8.     $nsfilename = $operatingsystem ? {
9.         Solaris => "/etc/nsswitch.conf",
10.        default => "/etc/nsswitch.conf"
11.    }
12.
13.    config_file { "$nsfilename":
14.        content => template("nsswitch/nsswitch.conf"),
15.    }
16. }

```

7.4.3 templates

Et maintenant le fichier de conf qui va faire appel au facter :



/etc/puppet/modules/nsswitch/templates/nsswitch.conf

```

|
|#
|# Copyright 2006 Sun Microsystems, Inc. All rights reserved.
|# Use is subject to license terms.
|#
|#
|# /etc/nsswitch.dns:
|#
|# An example file that could be copied over to /etc/nsswitch.conf; it uses
|# DNS for hosts lookups, otherwise it does not use any other naming service.
|#
|# "hosts:" and "services:" in this file are used only if the
|# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.
|#
|# DNS service expects that an instance of svc:/network/dns/client be
|# enabled and online.
|#
|passwd:      files ldap
|group:       files ldap
|
|# You must also set up the /etc/resolv.conf file for DNS name
|# server lookup. See resolv.conf(4).
|hosts:       files dns
|
|hosts:       <% if is_cluster -%>cluster<% end -%> files dns
|
|# Note that IPv4 addresses are searched for in all of the ipnodes databases
|# before searching the hosts databases.
|ipnodes:     files dns
|ipnodes:     files dns [TRYAGAIN=0]
|ipnodes:     files dns [TRYAGAIN=0 ]
|
|networks:    files
|protocols:   files
|rpc:         files
|ethers:      files
|netmasks:   files
|netmasks:   <% if is_cluster -%>cluster<% end -%> files
|bootparams:  files
|publickey:   files
|
|# At present there isn't a 'files' backend for netgroup; the system will
|# figure it out pretty quickly, and won't use netgroups at all.
|netgroup:    files
|automount:   files
|aliases:     files
|
|

```



```

services: files
printers:  user files
:
:
:
auth_attr: files
prof_attr: files
project:   files
:
:
:
tnrhttp:  files
tnrhdb:   files

```

7.5 Nagios NRPE + plugins

Vous devez initialiser le module avant de continuer.

7.5.1 init.pp

Imaginons que j'ai un dossier de plugins nagios que je souhaite déployer partout. Des fois j'en ajoute, j'en enlève, bref, je fais ma vie avec et je veux que ce soit pleinement synchronisé. Voici la marche à suivre :



/etc/puppet/modules/nrpe/manifests/init.pp

```

0. #nagios_plugins.pp
1. class nrpe {
2.     # Copy conf and send checks
3.     include nrpe::common
4.
5.     # Check operating system
6.     case $operatingsystem {
7.         Solaris: {
8.             include nrpe::service::solaris
9.         }
10.        default: { }
11.    }
12. }
13.
14. class nrpe::common {
15.     class nrpe::configs {
16.         # Used for nrpe-deimos.cfg templates
17.         $nrpe_distrib = $operatingsystem ? {
18.             'Solaris' => "/opt/csw/libexec/nagios-plugins",
19.             'redhat'  => "/home/deimos/checks/nrpe_scripts",
20.             'debian' => "/etc/nrpe",
21.         }
22.
23.         # Used for nrpe-deimos.cfg templates
24.         $deimos_script = $operatingsystem ? {
25.             'Solaris' => "/opt/deimos/libexec",
26.             'redhat'  => "/etc/nrpe.d",
27.             'debian' => "/etc/nrpe",
28.         }
29.
30.         # Copy NRPE config file
31.         file { ['opt/csw/etc/nrpe.cfg']:
32.             mode => 644, owner => root, group => root,
33.             content => template('nrpe/nrpe.cfg'),
34.         }
35.
36.         ## Copy and adapt NRPE deimos Config file ##
37.         file { ['opt/csw/etc/nrpe-deimos.cfg']:

```

```

38.         mode => 644, owner => root, group => root,
39.         content => template('nrpe/nrpe-deimos.cfg'),
40.     }
41. }
42.
43. class nrpe::copy_checks {
44.     ## Copy deimos Production NRPE Checks ##
45.
46.     file { "nrpe_prod_checks":
47.         name => $operatingsystem ? {
48.             'Solaris' => "/opt/deimos/libexec",
49.             'redhat' => "/home/deimos/checks/nrpe_scripts",
50.             'debian' => "/etc/nrpe",
51.         },
52.         ensure => directory,
53.         mode => 755, owner => root, group => root,
54.         source => [ "puppet:///modules/nrpe/Nagios/nrpechecks/deimos", "puppet:." ],
55.         recurse => true,
56.         force => true,
57.         ignore => '.svn'
58.     }
59. }
60.
61. include nrpe::configs
62. include nrpe::copy_checks
63. }
64.
65. class nrpe::service::solaris {
66.     ## Check service and restart if needed
67.     file { '/var/svc/manifest/network/nagios-nrpe.xml':
68.         source => "puppet:///modules/nrpe/nagios-nrpe.xml",
69.         mode => 644, owner => root, group => sys,
70.         before => Service["svc:/network/cswnrpe:default"]
71.     }
72.
73.     # Restart service if smf xml has changed
74.     exec { "`svccfg import /var/svc/manifest/network/nagios-nrpe.xml`" :
75.         subscribe => File['/var/svc/manifest/network/nagios-nrpe.xml'],
76.         refreshonly => true
77.     }
78.
79.     # Restart if one of those files have changed
80.     service { "svc:/network/cswnrpe:default":
81.         ensure => running,
82.         manifest => "/var/svc/manifest/network/nagios-nrpe.xml",
83.         subscribe => [ File['/opt/csw/etc/nrpe.cfg'], File['/opt/csw/etc/nrpe-deimo:
84.     ]
85. }

```

Ici :

- purge permet d'effacer les fichiers qui n'existent plus
- recurse : récursif
- force : permet de forcer
- before : permet d'être exécuter avant autre chose
- subscribe : inscrit une dépendance par rapport à la valeur de celui ci
- refreshonly : rafraichit seulement si il y a des changements

Avec le subscribe, vous pouvez le voir ici, on fait des listes de cette façon : [élément_1, élément_2,

élément_3...]. Par contre, petite précision, la changement opère (ici un restart) **seulement si l'un des éléments dans la liste est modifié et non tous**.

7.5.2 files

Pour la partie file, j'ai linké avec un svn externe qui permet de m'affranchir de l'intégration des plugins dans la partie puppet (qui entre nous n'a rien à faire ici).

7.5.3 templates

Il suffit de copier le dossier nagios_plugins dans files et de faire les templates ici (je ne ferais que le nrpe.cfg) :



/opt/csw/etc/nrpe.cfg

```
....
command[check_load]=<%= nrpe_distrib %>/check_load -w 15,10,5 -c 30,25,20
command[sunos_check_rss_mem]=<%= deimos_script %>/check_rss_mem.pl -w $ARG1$ -c $ARG2$
....
```

7.6 Munin

Disclaimer : this work is mostly based upon DavidS work, available on his [git repo (<http://git.black.co.at/>)]. In the scope of my work I needed to have munin support for freeBSD & Solaris. I also wrote a class for snmp_plugins & custom plugins. Some things are quite dependant from my infrastructure, like munin.conf generation script but it can easily be adapted to yours, by extracting data from your CMDB.

It requires the munin_interfaces fact published here (and merged into DavidS repo, thanks to him), and [Volcane's extlookup function (<http://nephilim.ml.org/~rip/puppet/extlookup.rb>)] to store some parameters. Enough talking, this is the code :



```
0. # Munin config class
1. # Many parts taken from David Schmitt's http://git.black.co.at/
2. # FreeBSD & Solaris + SNMP & custom plugins support by Nicolas Szalay <nico@gcu.inf>
3.
4. class munin::node {
5.     case $operatingsystem {
6.         openbsd: {}
7.         debian: { include munin::node::debian}
8.         freebsd: { include munin::node::freebsd}
9.         solaris: { include munin::node::solaris}
10.        default: {}
11.    }
12. }
13.
14. class munin::node::debian {
15.
16.     package { ["munin-node": ensure => installed ]
17.
18.     file {
19.         "/etc/munin":
20.             ensure => directory,
```

```
21.         mode => 0755,
22.         owner => root,
23.         group => root;
24.
25.     "/etc/munin/munin-node.conf":
26.         source => "puppet://$fileservers/files/apps/munin/munin-node-debian.",
27.         owner => root,
28.         group => root,
29.         mode => 0644,
30.         before => Package["munin-node"],
31.         notify => Service["munin-node"],
32.     }
33.
34.     service { "munin-node": ensure => running }
35.
36.     include munin::plugins::linux
37. }
38.
39. class munin::node::freebsd {
40.     package { "munin-node": ensure => installed, provider => freebsd }
41.
42.     file { "/usr/local/etc/munin/munin-node.conf":
43.         source => "puppet://$fileservers/files/apps/munin/munin-node-freebsd",
44.         owner => root,
45.         group => wheel,
46.         mode => 0644,
47.         before => Package["munin-node"],
48.         notify => Service["munin-node"],
49.     }
50.
51.     service { "munin-node": ensure => running }
52.
53.     include munin::plugins::freebsd
54. }
55.
56. class munin::node::solaris {
57.     # "hand made" install, no package.
58.     file { "/etc/munin/munin-node.conf":
59.         source => "puppet://$fileservers/files/apps/munin/munin-node-solaris",
60.         owner => root,
61.         group => root,
62.         mode => 0644
63.     }
64.
65.     include munin::plugins::solaris
66. }
67.
68. class munin::gatherer {
69.     package { "munin":
70.         ensure => installed
71.     }
72.
73.     # custom version of munin-graph : forks & generates many graphs in parallel
74.     file { "/usr/share/munin/munin-graph":
75.         owner => root,
76.         group => root,
77.         mode => 0755,
78.         source => "puppet://$fileservers/files/apps/munin/gatherer/munin-gra",
79.         require => Package["munin"]
80.     }
81.
82.     # custom version of debian cron file. Month & Year cron are generated once
83.     file { "/etc/cron.d/munin":
84.         owner => root,
85.         group => root,
86.         mode => 0644,
```

```

87.         source => "puppet://$filesERVER/files/apps/munin/gatherer/munin.cron"
88.         require => Package["munin"]
89.     }
90.
91.     # Ensure cron is running, to fetch every 5 minutes
92.     service { "cron":
93.         ensure => running
94.     }
95.
96.     # Ruby DBI for mysql
97.     package { "libdbd-mysql-ruby":
98.         ensure => installed
99.     }
100.
101.     # config generator
102.     file { "/opt/scripts/muningen.rb":
103.         owner => root,
104.         group => root,
105.         mode => 0755,
106.         source => "puppet://$filesERVER/files/apps/munin/gatherer/muningen."
107.         require => Package["munin", "libdbd-mysql-ruby"]
108.     }
109.
110.     # regenerate munin's gatherer config every hour
111.     cron { "munin_config":
112.         command => "/opt/scripts/muningen.rb > /etc/munin/munin.conf",
113.         user => "root",
114.         minute => "0",
115.         require => File["/opt/scripts/muningen.rb"]
116.     }
117.
118.     include munin::plugins::snmp
119.     include munin::plugins::linux
120.     include munin::plugins::custom::gatherer
121. }
122.
123.
124. # define to create a munin plugin inside the right directory
125. define munin::plugin ($ensure = "present") {
126.
127.     case $operatingsystem {
128.         freebsd: {
129.             $script_path = "/usr/local/share/munin/plugins"
130.             $plugins_dir = "/usr/local/etc/munin/plugins"
131.         }
132.         debian: {
133.             $script_path = "/usr/share/munin/plugins"
134.             $plugins_dir = "/etc/munin/plugins"
135.         }
136.         solaris: {
137.             $script_path = "/usr/local/munin/lib/plugins"
138.             $plugins_dir = "/etc/munin/plugins"
139.         }
140.         default: { }
141.     }
142.
143.     $plugin = "$plugins_dir/$name"
144.
145.     case $ensure {
146.         "absent": {
147.             debug ( "munin_plugin: suppressing $plugin" )
148.             file { $plugin: ensure => absent, }
149.         }
150.
151.         default: {
152.             $plugin_src = $ensure ? { "present" => $name, default => $e

```

```

153.
154.             file { $plugin:
155.                 ensure => "$script_path/${plugin_src}",
156.                 require => Package["munin-node"],
157.                 notify => Service["munin-node"],
158.             }
159.         }
160.     }
161. }
162.
163. # snmp plugin define, almost same as above
164. define munin::snmp_plugin ($ensure = "present") {
165.     $pluginname = get_plugin_name($name)
166.
167.     case $operatingsystem {
168.         freebsd: {
169.             $script_path = "/usr/local/share/munin/plugins"
170.             $plugins_dir = "/usr/local/etc/munin/plugins"
171.         }
172.         debian: {
173.             $script_path = "/usr/share/munin/plugins"
174.             $plugins_dir = "/etc/munin/plugins"
175.         }
176.         solaris: {
177.             $script_path = "/usr/local/munin/lib/plugins"
178.             $plugins_dir = "/etc/munin/plugins"
179.         }
180.         default: { }
181.     }
182.
183.     $plugin = "$plugins_dir/$name"
184.
185.     case $ensure {
186.         "absent": {
187.             debug ( "munin_plugin: suppressing $plugin" )
188.             file { $plugin: ensure => absent, }
189.         }
190.
191.         "present": {
192.             file { $plugin:
193.                 ensure => "$script_path/${pluginname}",
194.                 require => Package["munin-node"],
195.                 notify => Service["munin-node"],
196.             }
197.         }
198.     }
199. }
200.
201. class munin::plugins::base
202. {
203.     case $operatingsystem {
204.         debian: { $plugins_dir = "/etc/munin/plugins" }
205.         freebsd: { $plugins_dir = "/usr/local/etc/munin/plugins" }
206.         solaris: { $plugins_dir = "/etc/munin/plugins" }
207.         default: {}
208.     }
209.
210.     file { $plugins_dir:
211.         source => "puppet://$fileserver/files/empty",
212.         ensure => directory,
213.         checksum => mtime,
214.         ignore => ".svn*",
215.         mode => 0755,
216.         recurse => true,
217.         purge => true,
218.         force => true,

```

```

219.         owner => root
220.     }
221. }
222.
223. class munin::plugins::interfaces
224. {
225.     $ifs = gsub(split($munin_interfaces, " "), "(.+)", "if_\\1")
226.     $if_errs = gsub(split($munin_interfaces, " "), "(.+)", "if_err_\\1")
227.     plugin {
228.         $ifs: ensure => "if_";
229.         $if_errs: ensure => "if_err_";
230.     }
231.
232.     include munin::plugins::base
233. }
234.
235. class munin::plugins::linux
236. {
237.     plugin { [ cpu, load, memory, swap, irq_stats, df, processes, open_files, n
238.         ensure => "present"
239.     }
240.
241.     include munin::plugins::base
242.     include munin::plugins::interfaces
243. }
244.
245. class munin::plugins::nfsclient
246. {
247.     plugin { "nfs_client":
248.         ensure => present
249.     }
250. }
251.
252. class munin::plugins::snmp
253. {
254.     # initialize plugins
255.     $snmp_plugins=extlookup("munin_snmp_plugins")
256.     snmp_plugin { $snmp_plugins:
257.         ensure => present
258.     }
259.
260.     # SNMP communities used by plugins
261.     file { "/etc/munin/plugin-conf.d/snmp_communities":
262.         owner => root,
263.         group => root,
264.         mode => 0644,
265.         source => "puppet://$fileserver/files/apps/munin/gatherer/snmp_comm
266.     }
267.
268. }
269.
270. define munin::custom_plugin($ensure = "present", $location = "/etc/munin/plugins") {
271.     $plugin = "$location/$name"
272.
273.     case $ensure {
274.         "absent": {
275.             file { $plugin: ensure => absent, }
276.         }
277.
278.         "present": {
279.             file { $plugin:
280.                 owner => root,
281.                 mode => 0755,
282.                 source => "puppet://$fileserver/files/apps/munin/cu
283.                 require => Package["munin-node"],
284.                 notify => Service["munin-node"],

```

```

285.         }
286.     }
287. }
288. }
289.
290. class munin::plugins::custom::gatherer
291. {
292.     $plugins=extlookup("munin_custom_plugins")
293.     custom_plugin { $plugins:
294.         ensure => present
295.     }
296. }
297.
298. class munin::plugins::freebsd
299. {
300.     plugin { [ cpu, load, memory, swap, irq_stats, df, processes, open_files, n
301.         ensure => "present",
302.     }
303.
304.     include munin::plugins::base
305.     include munin::plugins::interfaces
306. }
307.
308. class munin::plugins::solaris
309. {
310.     # Munin plugins on solaris are quite ... buggy. Will need rewrite / custom
311.     plugin { [ cpu, load, netstat ]:
312.         ensure => "present",
313.     }
314.
315.     include munin::plugins::base
316.     include munin::plugins::interfaces
317. }

```

7.6.1 Munin Interfaces

Everyone using puppet knows DavidS awesome git repository : git.black.co.at. Unfortunately for me, his puppet infrastructure seems to be almost only linux based. I have different OS in mine, including FreeBSD & OpenSolaris. Looking at his module-munin I decided to reuse it (and not recreate the wheel) but he used a custom fact that needed some little work. So this is a FreeBSD & (Open)Solaris capable version, to know what network interfaces have link up.



```

0. # return the set of active interfaces as an array
1. # taken from http://git.black.co.at
2. # modified by nico <nico@gcu.info> to add FreeBSD & Solaris support
3.
4. Factor.add("munin_interfaces") do
5.
6.     setcode do
7.         # linux
8.         if Factor.value('kernel') == "Linux" then
9.             `ip -o link show`.split(/\n/).collect do |line|
10.                value = nil
11.                matches = line.match(/^d*: ([^:]*): <(.*),'

```



```

12.             if !matches.nil?
13.                 value = matches[1]
14.                 value.gsub!(/@.*/, '')
15.             end
16.             value
17.         end.compact.sort.join(" ")
18.     #end
19.
20.     # freebsd
21.     elsif Facter.value('kernel') == "FreeBSD" then
22.         Facter.value('interfaces').split(/,/).collect do |interface
23.             status = `ifconfig #{interface} | grep status`
24.             if status != "" then
25.                 status=status.strip!.split(":")[1].strip!
26.                 if status == "active" then # I CAN HAZ LINK
27.                     interface.to_a
28.                 end
29.             end
30.         end.compact.sort.join(" ")
31.     #end
32.
33.     # solaris
34.     elsif Facter.value('kernel') == "SunOS" then
35.         Facter.value('interfaces').split(/,/).collect do |interface
36.             if interface != "lo0" then # /dev/lo0 does not exist
37.                 status = `ndd -get /dev/#{interface} link_up`
38.                 if status == "1" # ndd returns 1 for link up
39.                     interface.to_a
40.                 end
41.             end
42.         end.compact.sort.join(" ")
43.     end
44. end
45. end

```

7.7 Importation d'un module

Les modules doivent être importés dans puppet pour qu'ils soient pris en charge. Ici, nous n'avons pas à nous en soucier puisque d'après la configuration serveur que nous en avons faite, il va automatiquement charger les modules dans /etc/puppet/modules. Cependant si vous souhaitez autoriser module par module, il vous faut les importer à la main :



/etc/puppet/manifests/modules.pp

```


0. # /etc/puppet/manifests/modules.pp
1.
2. import "sudo"
3. import "ssh"

```

Faites attention car à partir du moment où ceci est renseigné, nul besoin de redémarrer puppet pour que les changements soient pris en compte. Il va donc falloir faire attention à ce que vous rendez disponible à l'instant t.

7.7.1 Importer tous les modules d'un coup

Ceci peut avoir de graves conséquences, mais sachez qu'il est possible de le faire :

 /etc/puppet/manifests/modules.pp


```
0. # /etc/puppet/manifests/modules.pp
1.
2. import "*.pp"
```

8 Configuration avancée

8.1 Création de Facts

Les "facts", sont des scripts (voir /usr/lib/ruby/1.8/facter pour les facts standards) permettant de construire des variables dynamiques, qui changent en fonction de l'environnement dans lequel ils sont exécutés.

Par exemple, on pourra définir un "fact", qui détermine si l'on est sur une machine de type "cluster" en fonction de la présence ou l'absence d'un fichier :

 is_cluster.rb

```
0. # is_cluster.rb
1.
2. Facter.add("is_cluster") do
3.   setcode do
4.     FileTest.exists?("/etc/cluster/nodeid")
5.   end
6. end
```

Attention, si l'on veut tester le fact sur la machine destination, il ne faudra pas oublier de spécifier le chemin où se trouvent les facts sur la machine :

 export

```
export FACTERLIB=/var/lib/puppet/lib/facter
```

ou pour Solaris :

 export

```
export FACTERLIB=/var/opt/csw/puppet/lib/facter
```

Pour voir la liste des facts ensuite présent sur le système, il faut simplement taper la commande facter :

 facter

```
0. > facter
1. facterversion => 1.5.7
2. hardwareisa => i386
3. hardwaremodel => i86pc
4. hostname => PA-OFC-SRV-UAT-2
5. hostnameldap => PA-OFC-SRV-UAT
6. id => root
7. interfaces => lo0,e1000g0,e1000g0_1,e1000g0_2,e1000g1,e1000g2,e1000g3,clprivnet0
8. ...
```

Voir http://projects.puppetlabs.com/projects/puppet/wiki/Adding_Facts pour plus de détails.

8.2 Outrepasser des restrictions

Si par exemple, nous avons défini une classe et que pour certains hôtes, nous souhaitons modifier cette configuration, nous devons faire comme ceci :



```
0. class somehost_postfix inherits postfix {
1.     # blah blah blah
2. }
3.
4. node somehost {
5.     include somehost_postfix
6. }
```

Admettons que nous avons un module postfix de défini. Nous souhaitons appliquer une config particulière à certains hosts défini ici par 'somehost'. Pour bypasser la configuration, il faut créer une classe somehost_postfix'. J'insiste ici sur la nomenclature du nom à donner pour cette classe puisque c'est uniquement comme cela que Puppet reconnaîtra que vous souhaitez appliquer une configuration particulière.

8.3 Puppet Push

Puppet fonctionne en mode client -> serveur. C'est le client qui contacte toutes les 30 min (par défaut) le serveur et demande une synchronisation. Lorsque vous êtes en mode boulet ou bien quand vous souhaitez déclencher à un instant t la mise à jour de vos machine clientes vers le serveur, il y a un mode particulier (listen). Le problème c'est que le client puppet ne peut pas tourner en mode client et listen. Il faut donc 2 instances...bref les cauchemars commencent.

Pour palier à ce problème, j'ai donc développé Puppet push qui permet de demander aux client (via SSH) de se synchroniser. Vous l'aurez compris, il est plus que nécessaire d'avoir un échange de clef effectué au préalable avec les clients. Comment faire ? Pas de soucis, nous avons vu cela plus haut.

On peut faire pas mal de choses avec Puppet push, c'est pourquoi il devient assez pratique et utilisé lorsqu'on commence à jouer avec. Aujourd'hui je suis en version 0.1b et ai pas mal d'idées pour le faire évoluer. Pour accéder à la dernière version, suivez ce lien : http://www.deimos.fr/gitweb/?p=puppet_push.git;a=tree

9 FAQ

9.1 err: Could not retrieve catalog from remote server: hostname was not match with the server certificate

Vous avez un problème avec vos certificats. Le mieux c'est de régénérer un certificats avec tous les hostname du serveur dans ce certificat : Création d'un certificat

10 Ressources

<http://reductivelabs.com/products/puppet/>

<http://www.rottenbytes.info>

Modules pour Puppet (<http://git.black.co.at/>)

Puppet recipes (<http://reductivelabs.com/trac/puppet/wiki/Recipes>)

Types d'objets pour Puppet (<http://reductivelabs.com/trac/puppet/wiki/TypeReference>)

Puppet SSL Explained (<http://www.masterzen.fr/2010/11/14/puppet-ssl-explained/>) (PDF)

Récupérée de « <http://www.deimos.fr/blocnotesinfo>

[/index.php?title=Puppet:_Solution_de_gestion_de_fichier_de_configuration&oldid=9796](http://www.deimos.fr/blocnotesinfo/index.php?title=Puppet:_Solution_de_gestion_de_fichier_de_configuration&oldid=9796) »
