Puppet par inesmk et stephan

1. Présentation

Puppet est un outil Open source qui permet l'automatisation de la gestion et la configuration des infrastructures. Il facilite ainsi le travail de l'administrateur en réduisant le temps alloué aux tâches répétitives (installation de la même application sur plusieurs machines, création des utilisateurs…etc). En effet, il permet le télédéploiement de configuration sur un ensemble de serveurs en quelques minutes. Par exemple Puppet permet d'automatiser l'installation de logiciels, de services ou encore de modifier des fichiers et 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. L'intérêt de cette solution open source réside dans son support multi-plateformes (basé sur Ruby), sa sécurité (ssl), son développement actif et sa relative simplicité à mettre en oeuvre. Il est à noter que puppet est une évolution de cfengine.

a) Pourquoi la gestion centralisée ?

La gestion centralisée facilite le travail de l'administrateur système et la supervision des machines et ressources. En effet, Puppet peut s'avérer un outil puissant pour pallier aux difficultés suivantes :

- Dans le cas d'un grand nombre de serveurs → il est difficile d'avoir une vue d'ensemble de ce qu'il y a sur chaque système,
- Remplacer efficacement un serveur défectueux ou rajouter un autre serveur identique,
- Garantir une configuration identique sur plusieurs serveurs,
- Réaliser efficacement des changements sur tous les serveurs qui ont besoin d'un changement particulier.
- Arrêter les taches répétitives et les efforts de duplication.

b) Pourquoi Puppet ?

Il existe plusieurs outils qui proposent le même type de solution qui sont Chef et Cfengine (pour plus d'information vous pouvez consultez sur http://verticalsysadmin.com/blog/relative-origins-of-cfengine-chef-and-puppet)

Puppet présente un certain nombre d'avantage :

- Puppet fonctionne sur la plupart des systèmes Uni* et dans une moindre mesure Windows (Windows est officiellement supporté depuis la version 2.7.x),
- puppet est l'outils qui se rapproche le plus (parmi les outils existants) de la conception “humaine” de la configuration “. Le langage que puppet utilise est specifiquement conçu pour cet objectif. Donc, il est optimisé pour la description et le lien entre les ressources tels que les utilisateurs ou les fichiers.
- puppet utilise SSL (https://fr.wikipedia.org/wiki/Transport_Layer_Security) pour communiquer d'une façon sécurisé entre le client et le serveur,
- puppet a une communauté large et active qui développe et supporte activement le produit (plus de 80 organisation où on peut citer Google, Red Hat, Siemens, au niveau des entreprises at Stanford and Harvard Law School au niveau des universités).
- Puppet est un outil solide. Il a déjà été déployé sur de très grandes infrastructures (+ de 5000 machines).
- Puppet dispose d'une documentation très riche et fournie tout aussi bien sur le langage que sur les ressources.

Puppet peut gérer des objets chez le client appelée ressources. Les ressources sont de différents types : cron, file, group, host, package, service, user… Puppet installe automatiquement les outils appropriés en fonction des fournisseurs et des plateformes correspondantes. Par exemple, le management des packets est réalise par apt-get sur debian et par yum sur Centos.

c) Les différentes versions et distributions de Puppet

L'outil puppet a différentes versions, une version libre puppet et une version commerciale puppet entreprise. La version entreprise dispose d'une interface graphique de gestion “Dashbord”.

Les différentes version de puppet sont listées sur https://rubygems.org/gems/puppet/versions/. La première version 0.9.2 date de November 22, 2005. et la version la plus récente est la version 4.6.2 datée de September 2, 2016

Par ailleurs, les versions de puppet Entreprise débutent à la version 2.7. La version la plus récente est Puppet Enterprise 2016.1. Cette numérotation s'explique par le fait que depuis juillet 2015, Puppet entreprise est passé à une nouveau système de visionnage. Les nouvelles version suivent un modèle “x.y.z” où “x” reflète l'année de sortie, “y” reflète le numéro de sortie de l'année en cours et “z” reflète la sortie du patch/bugfix. Ainsi, la première sortie de cette série a été Puppet Entreprise 2015.2. suivie par le patch 2015.2.1. Ainsi, les versions PE 3.3 to PE 3.7 n'ont pas existé afin de s'adapter à la nouvelle numérotation des versions (Il n'y a pas de PE 3.4, PE 3.5, or PE 3.6).

d) Le fonctionnement général de Puppet

Dans puppet on définit :

- client (Agent) est puppeted
- serveur est puppetmasterd

Les agents démarrent l'application puppet agent, généralement comme un service en arrière plan. Si Le puppet master/serveur est également démarré alors Le fonctionnement se déroule comme suit:

- Périodiquement, le client (agent) se connecte au serveur (généralement toute les 30 min envoi des facts au puppet master et demande un catalogue,
- le serveur compile la configuration pour le client et le lui envoi,
- le client vérifie que toutes les ressources mentionnées dans le catalogue qu'il a reçu. Il vérifie la conformité avec l'état de la machine. Il effectue toute les modifications nécessaire pour corriger les erreurs éventuelles.
- une fois qu'il a fini d'appliquer le catalogue, l'agent soumet un rapport au puppet master.

La configuration du client (node) peut être stockée dans le LDAP (https://fr.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol) ou dans une base de donnée.

Pour fonctionner, Puppet a besoin de :

- ruby, qui est un langage de programmation,
- Puppet, l'outil lui-même dans sa version master et agent,

On peut aussi utiliser Facter. C'est un outil qui permet de réunir les informations relative aux clients. Ces informations peuvent être utilisées comme des variables dans puppet. En effet, Facter permet de modifier les facts au besoin. Un fact peut contenir des données sur l'architecture, la version du OS, le domaine, la version de facter,.... Il est possible de voir la liste des paramètres du système qui sont automatiquement réunis dans le facter avec la commande /opt/puppetlabs/bin/facter

2. Installation sous CentOS 6

Sous cette version, le fonctionnement de puppet requiert l'ensemble passenger/apache et puppetmaster. Dans la version 7, cet ensemble a été remplacé par puppet serveur.

a) Installation du puppetmaster

Vérifier la version de Centos

```
[root@puppet ~]# cat /etc/redhat-release
CentOS release 6.6 (Final)
```

Changer le nom du domaine

```
[root@puppet ~]# vi /etc/sysconfig/network
```

```
ETWORKING=yes
HOSTNAME=puppet.fenestros.loc
```

Installer ntp (Network time protocol) afin d'avoir un temps de référence cohérent entre les différentes machines

```
[root@puppet ~]# yum install ntp
```

```
Loaded plugins: fastestmirror, refresh-packagekit, security
Setting up Install Process
...
Complete!
```

Sur les versions RH6 et antérieure il est nécessaire de rebooter le système afin que l'installation et la mise en place de ntp soit effective(ceci n'est plus nécessaire à partir de la version 7)

```
[root@puppet ~]# reboot
```

Installer puppet entreprise linux 6

```
sudo rpm -ivh https://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
```

le depot sera strocké dans /etc/yum.repos.d

```
[root@puppet ~]# cd /etc/yum.repos.d
[root@puppet yum.repos.d]# ls
adobe-linux-i386.repo  CentOS-Media.repo  mirrors-rpmforge
atomic.repo            CentOS-Vault.repo  mirrors-rpmforge-extras
CentOS-Base.repo       chromium-el6.repo  mirrors-rpmforge-testing
CentOS-Debuginfo.repo  epel.repo          puppetlabs.repo
CentOS-fasttrack.repo  epel-testing.repo  rpmforge.repo
```

Il est recommandé de vider le cache régulièrement. Cette opération n'est pas nécessaire mais ça peut être une bonne pratique à acquérir.

```
[root@puppet ~]# yum clean all
Loaded plugins: fastestmirror, refresh-packagekit, security
...
Cleaning up list of fastest mirrors
```

Installation de puppet server

```
[root@puppet ~]# yum install puppet-server
```

Configuration du puppetmaster

La configuration de puppet se trouve dans /etc/puppet/puppet.conf.

Configurer la FQDN (Fully qualified domain name) du serveur

```
[root@puppet ~]# vi /etc/puppet/puppet.conf
```

Rajouter la ligne suivante (et son commentaire) dans la section [main] du fichier puppet.conf

```
#configurer un nom DNS auquel le serveur devra repondre
    dns_alt_names = puppet puppet.fenestros.loc
```

établir une communication sécurisée entre le client et le serveur.

```
[root@puppet ~]# puppet master --verbose --no-daemonize
```

Dès l'apparition du message “Notice: Starting Puppet master version 3.8.7” pensez à arrêter le processus (à cette étape le client n'a pas encore été configuré donc la communication ne peut pas être établie)

```
[root@puppet environments]# vi /etc/puppet/puppet.conf
```

Rajouter la ligne (commentée) suivante dans le fichier puppet.config se trouvant dans /etc/puppet/

```
#indiquer à puppet la localisation des repertoires correspondants qux environnement
    environmentpath = $confdir/environments
```

Démarrer puppet

```
[root@puppet environments]# service puppetmaster start
Starting puppetmaster:                                     [  OK  ]
```

Une fois assuré qu'il fonctionne correctement, on va l’arrêter puisque nous n'en avons pas besoin pour le moment.

```
[root@puppet environments]# service puppetmaster stop
Stopping puppetmaster:                                     [  OK  ]
```

Installation d'Appache et des outils nécessaires au fonctionnement de puppet

```
[root@puppet environments]# yum install httpd httpd-devel mod_ssl ruby-devel rubygems gcc
[root@puppet environments]# chkconfig on
```

le système requière également ruby 2.2.2. Or pour installer ce langage, les développeurs Ruby ont conçu un outil spécifique RVM (Ruby Version Manager). Pour installer RVM

```
[root@puppet ~]# yum install gcc-c++ patch readline readline-devel zlib zlib-devel
[root@puppet ~]# yum install libyaml-devel libffi-devel openssl-devel make
[root@puppet ~]# yum install bzip2 autoconf automake libtool bison iconv-devel sqlite-devel
[root@puppet ~]# curl -sSL https://rvm.io/mpapis.asc | gpg --import -
[root@puppet ~]# curl -L get.rvm.io | bash -s stable
[root@puppet ~]# source /etc/profile.d/rvm.sh
[root@puppet ~]# rvm reload
[root@puppet ~]# rvm requirements run
Checking requirements for centos.
Requirements installation successful.
[root@puppet ~]# rvm install 2.2.2
Install of ruby-2.2.2 - #complete
```

Installer passenger Passenger est l'agent de liaison

```
[root@puppet~]# gem install rack passenger
Fetching: rack-2.0.1.gem (100%)
Successfully installed rack-2.0.1
...
Done installing documentation for passenger after 87 seconds
2 gems installed
[root@puppet ~]# passenger-install-apache2-module
```

Vous allez vous retrouver dans des fenêtres d'installation, appuyez sur entrée de façon successives et laissez le choix par défaut du langage Ruby. Il est possible que certains paquets nécessaire au fonctionnement de la machine soient manquants. L'outil vous l'indiquera et vous indiquera comment faire pour les récupérer exemple :

```
Some required software is not installed.
But don't worry, this installer will tell you how to install them.
Press Enter to continue, or Ctrl-C to abort.
--------------------------------------------
Installation instructions for required software
 * To install Curl development headers with SSL support:
   Please install it with yum install libcurl-devel
```

Une fois que l'outil dispose de tous les paquets nécessaires il faut relancer l'installation de passenger

```
[root@puppet ~]# passenger-install-apache2-module
```

Cette opération peut prendre quelques minutes. Appuyiez sur entrée jusqu'à retrouver la ligne de commande.

Maintenant il faut configurer Apache pour utiliser passenger

```
[root@puppet ~]# vi /etc/httpd/conf.d/puppet.conf
```

copiez le fichier si dessus :

```
# Debian/Ubuntu:
#LoadModule passenger_module /var/lib/gems/1.8/gems/passenger-4.0.x/ext/apache2/mod_passenger.so
#PassengerRoot /var/lib/gems/1.8/gems/passenger-4.0.x
#PassengerRuby /usr/bin/ruby1.8

# RHEL/CentOS:
LoadModule passenger_module /usr/local/rvm/gems/ruby-2.2.2/gems/passenger-5.0.30/buildout/apache2/
mod_passenger.so
PassengerRoot /usr/local/rvm/gems/ruby-2.2.2/gems/passenger-5.0.30
PassengerRuby /usr/bin/ruby

# And the passenger performance tuning settings:
# Set this to about 1.5 times the number of CPU cores in your master:
PassengerMaxPoolSize 12
```

```
# Recycle master processes after they service 1000 requests
PassengerMaxRequests 1000
# Stop processes if they sit idle for 10 minutes
PassengerPoolIdleTime 600

Listen 8140
<VirtualHost *:8140>
    # Make Apache hand off HTTP requests to Puppet earlier, at the cost of
    # interfering with mod_proxy, mod_rewrite, etc. See note below.
    PassengerHighPerformance On

    SSLEngine On

    # Only allow high security cryptography. Alter if needed for compatibility.
    SSLProtocol ALL -SSLv2 -SSLv3
    SSLCipherSuite
EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+AESGCM:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EECDH:+CAMELLIA256:+AES256:+CAMELLIA1
28:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:!IDEA:!ECDSA:kEDH:CAMELLIA256-
SHA:AES256-SHA:CAMELLIA128-SHA:AES128-SHA
    SSLHonorCipherOrder     on

    SSLCertificateFile      /var/lib/puppet/ssl/certs/puppet.fenestros.loc.pem
    SSLCertificateKeyFile   /var/lib/puppet/ssl/private_keys/puppet.fenestros.loc.pem
    SSLCertificateChainFile /var/lib/puppett/ssl/ca/ca_crt.pem
    SSLCACertificateFile    /var/lib/puppet/ssl/ca/ca_crt.pem
    SSLCARevocationFile     /var/lib/puppet/ssl/ca/ca_crl.pem
   #SSLCARevocationCheck    chain
    SSLVerifyClient         optional
    SSLVerifyDepth          1
    SSLOptions              +StdEnvVars +ExportCertData

    # Apache 2.4 introduces the SSLCARevocationCheck directive and sets it to none
    # which effectively disables CRL checking. If you are using Apache 2.4+ you must
    # specify 'SSLCARevocationCheck chain' to actually use the CRL.
```

```
    # These request headers are used to pass the client certificate
    # authentication information on to the Puppet master process
    RequestHeader set X-SSL-Subject %{SSL_CLIENT_S_DN}e
    RequestHeader set X-Client-DN %{SSL_CLIENT_S_DN}e
    RequestHeader set X-Client-Verify %{SSL_CLIENT_VERIFY}e

    DocumentRoot /usr/share/puppet/rack/puppetmasterd/public

    <Directory /usr/share/puppet/rack/puppetmasterd/>
      Options None
      AllowOverride None
      # Apply the right behavior depending on Apache version.
      <IfVersion < 2.4>
        Order allow,deny
        Allow from all
      </IfVersion>
      <IfVersion >= 2.4>
        Require all granted
      </IfVersion>
    </Directory>

    ErrorLog /var/log/httpd/puppet-server.example.com_ssl_error.log
    CustomLog /var/log/httpd/puppet-server.example.com_ssl_access.log combined
</VirtualHost>
```

Modifier le fichier principal de configuration de Apache

```
[root@puppet ~]# vi /etc/httpd/conf/httpd.conf
```

Chercher ServerName dans le fichier httpd.conf et de-commentez la ligne en remplaçant l'exemple par le nom du domaine que vous aviez choisi au début de l'installation

```
ServerName puppet.fenestros.loc
```

b) installation de de l'agent

Importer une nouvelle machine qui sera l'agent géré par le puppet master. Changer le nom du domaine

```
[root@Centos6 ~]# vi /etc/sysconfig/network
```

```
ETWORKING=yes
HOSTNAME=agent.fenestros.loc
```

La première chose à faire est de s'assurer que l'agent peut communiquer avec le puppet master

```
[root@agent ~]# vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain localhost
::1             localhost6.localdomain6 localhost6
10.0.2.16       puppet.fenestros.loc
10.0.2.15       agent.fenestros.loc
```

[root@agent ~]# ping puppet.fenestros.loc</code>

Recuperation des paquets du puppet agent

```
[root@agent ~]# rpm -fvh http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm

[root@agent ~]# yum install puppet
Complete!
```

```
[root@agent ~]# vim /etc/puppet/puppet.conf
```

Rajouter deux lignes dans [main]:

```
# Set the server that the agent will get config setting from (puppet Master)
server = puppet.fenestros.loc
```

```
# Turn on reporting
report = true
```

```
[root@agent ~]# service puppet start
starting puppet agent:                                     [ok]
[root@agent ~]# check config puppet on
```

revenir sur la machine serveur et approuver la certification

```
[root@puppet ~]# sudo puppet cert list
[root@puppet ~]# sudo puppet cert sign agent
```

3. Installation sous CentOS 7

a) Installation du serveur

Configuration d'une adresse IPfixe pour le puppet master

```
[root@centos7 ~]# nmcli connection add con-name ip_fixe ifname enp0s3 type ethernet ip4 10.0.2.16/24 gw4 10.0.2.2
```

```
[root@centos7 ~]# vi /etc/hostname
```

```
puppet.fenestrotros.loc
```

```
[root@puppet ~]# nmcli connection mod ip_fixe ipv4.dns 8.8.8.8
[root@puppet ~]# vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain localhost
```

```
::1             localhost6.localdomain6 localhost6
10.0.2.15       node.fenestros.loc
10.0.2.16       puppet.fenestros.loc
```

```
[root@puppet ~]# systemctl restart NetworkManager
[root@puppet ~]# systemctl status NetworkManager
● NetworkManager.service - Network Manager
   Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2016-09-08 16:31:07 CEST; 12s ago
```

Il faut s'assurer que le serveur et le client utilise me même fuseau horaire; d'ou l'installation du serveur NTP

```
[root@puppet ~]# yum -y install ntpdate
Complete!
```

Installation de puppet à partir du depot approrié

```
[root@puppet ~]# rpm -Uvh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
Retrieving https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
warning: /var/tmp/rpm-tmp.nPW7FW: Header V4 RSA/SHA1 Signature, key ID 4bd6ec30: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:puppetlabs-release-pc1-1.0.0-2.el################################# [100%]
```

```
[root@puppet ~]# yum install -y puppetserver
Complete!
```

Activer puppetserver afin qu'il démarre au redémarrage du master-serveur. Pour le moment le master serveur ne gère aucun agent node.

```
[root@puppet ~]#systemctl enable puppetserver
```

b) Installation de l'agent

Configuration d'une adresse IPfixe pour le puppet agent Importez une nouvelle machine virtuelle Centos 7

```
[root@centos7 ~]# nmcli connection add con-name ip_fixe ifname enp0s3 type ethernet ip4 10.0.2.15/24 gw4 10.0.2.2
```

```
[root@centos7 ~]# vi /etc/hostname
```

```
node.fenestrotros.loc
```

```
[root@centos7 ~]# nmcli connection mod ip_fixe ipv4.dns 8.8.8.8
[root@node ~]# vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain localhost
::1             localhost6.localdomain6 localhost6
10.0.2.15       node.fenestros.loc
10.0.2.16       puppet.fenestros.loc
```

```
[root@node ~]# systemctl restart NetworkManager
[root@node ~]# systemctl status NetworkManager
● NetworkManager.service - Network Manager
   Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2016-09-08 19:39:45 CEST; 3min 59s ago
```

Installer le même fuseau horaire que le serveur afin de pouvoir être sur que les deux machines sont sur le même fuseau horaire.

```
[root@node ~]## yum -y install ntpdate
... Complete!
```

Recupération du dépôt

```
[root@node ~]# rpm -Uvh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
Retrieving https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm
```

```
warning: /var/tmp/rpm-tmp.LuoSjx: Header V4 RSA/SHA1 Signature, key ID 4bd6ec30: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:puppetlabs-release-pc1-1.0.0-2.el################################# [100%]
[root@node ~]#yum install -y puppet-agent
...Complete!
```

c) Signature des certificats

La première fois que puppet contrôle un agent node, il envoie une requête de signature de certificat au puppet master. Avant même que le puppet serveur soit capable de communiquer et controler l'agent node, il doit signer ce certificat.

Signer les demandes de certificats courants :

```
[root@puppet ~]# /opt/puppetlabs/bin/puppet cert sign node.fenestros.loc
```

Signer toutes les demandes de certificats :

```
[root@puppet ~]# /opt/puppetlabs/bin/puppet cert sign --all
```

Voir toute la liste de toutes les demandes de certification signée et non signée

```
[root@puppet ~]# /opt/puppetlabs/bin/puppet cert list --all
```

Il est à noté que les requêtes signée sont précédées par un +.

4. Configuration de Base

a) Configuration du Serveur

Il faut d'abord définir la quantité de mémoire à utiliser en éditant le fichier puppetserver (la valeur par défaut est de 2G).

```
 nano /etc/sysconfig/puppetserver
......
JAVA_ARGS="-Xms2G -Xmx2G -XX:MaxPermSize= 512m"
.....
```

Le fichier de configuration du serveur est dans /etc/puppetlabs/puppet/puppet.conf

éditer le et ajouter les lignes ci dessous nano /etc/puppetlabs/puppet/puppet.conf

```
[master]
.....
dns_alt_names = puppet.fenestros.loc,puppet
[main]
certname = puppet.fenestros.loc
server = puppet.fenestros.loc
environment = production
runinterval = 1h
Ensuite, démarrer le serveur et activer le
# systemctl start puppetserver
# systemctl enable puppetserver
Enfin ouvrir le port 8140 par lequel puppetMaster écoute
# firewall-cmd --permanent --zone=public --add-port=8140/tcp
# firewall-cmd --reload
```

b) Configuration de l'agent puppet

Le fichier de configuration de l'agent est dans /etc/puppetlabs/puppet/puppet.conf éditer le et ajouter les lignes ci dessous nano /etc/puppetlabs/puppet/puppet.conf

```
[main]
certname = node.fenestros.loc
```

```
server = puppet.fenestros.loc
environment = production
runinterval = 1h
```

Le runinterval permet de spécifier, l'intervalle entre deux connexions que fera le client, vers le serveur. Exécuter la commande suivante pour démarrer l'agent et pour que le démarrage se fasse automatiquement

```
# /opt/puppetlabs/bin/puppet resource service puppet ensure=running enable=true

Vous obtiendrez ceci apres execution
Notice: /Service[puppet]/ensure: ensure changed 'stopped' to 'running'
service { 'puppet':
  ensure => 'running',
  enable => 'true',
}
```

Le serveur le l'agent utilisant une connexion sécurisée, le client doit avoir un certificat fourni pas le client Exécuter cette commande sur le serveur pour connaitre la liste des certificats

```
# /opt/puppetlabs/bin/puppet cert list
```

Ensuite executer cette commande pour creer un certificat pour l'agent

```
# /opt/puppetlabs/bin/puppet cert sign node.fenestros.loc
```

5. LAB #1 - Utilisation

Puppet peut gérer plusieurs environnements, et il est possible de le faire en définissant des manifestes (ensembles de règles appliquées à un environnement). Puppet utilise un langage domaine-spécifique pour décrire les configuration des systèmes. Toutes ces descriptions sont enregistrées dans des manifestes qui ont une extension .pp. Il existe un répertoire par défaut, dans le master-serveur, pour stocker les manifestes au /etc/puppetlabs/code/environments/production/manifests/ Le manifeste principal est vide ⇒ Puppet n’exécutera aucune configuration de l'agent node. Nous allons créer un fichier site.pp dans le répertoire manifestes:

```
[root@puppet ~] # touch /etc/puppetlabs/code/environments/production/manifests/site.pp
[root@puppet ~]# ls -l /etc/puppetlabs/code/environments/production/manifests/
total 0
-rw-r--r--. 1 root root 0 Sep  8 22:54 site.pp
```

a) Création/modification d'un manifeste

au niveau du serveur

```
[root@puppet ~]# vi /etc/puppetlabs/code/environments/production/manifests/site.pp
```

```
file {'/tmp/example-ip':                                          # resource type file and filename
  ensure  => present,                                             # make sure it exists
  mode    => '0644',                                              # file permissions
  content => "Here is my Public IP Address: ${ipaddress_eth0}.\n",  # note the ipaddress_eth0 fact
}
```

ce manifeste doit assurer le fait que tous les agents nodes vont avoir un fichier localisé dans /tmp/example-ip avec des permissions-rw-r–r–. Avec un contenu qui inclu l'address IP publique du node.

au niveau de l'agent node

```
[root@node ~]# puppet agent --test
[root@node ~]# cat /tmp/example-ip
```

b) Création/modification d'un module

Les modules sont utiles pour regrouper les taches ensemble. Il existe plusieurs modules disponibles dans la communauté puppet.

au niveau du serveur puppet Dans le serveur-puppet on va installer le module puppetslabs-apache

```
[root@puppet ~]# /opt/puppetlabs/bin/puppet module install puppetlabs-apache
Notice: Preparing to install into /etc/puppetlabs/code/environments/production/modules ...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppetlabs/code/environments/production/modules
└─┬ puppetlabs-apache (v1.10.0)
  ├── puppetlabs-concat (v2.2.0)
  └── puppetlabs-stdlib (v4.12.0)
```

Attention : n'utilisez pas ce module avec une installation prealable d'apache. Il purgera toutes les configurations apache non gerée par puppet

```
[root@puppet ~]# vi /etc/puppetlabs/code/environments/production/manifests/site.pp
```

```
node 'host2' {
  class { 'apache': }             # use apache module
  apache::vhost { 'example.com':  # define vhost resource
    port    => '80',
    docroot => '/var/www/html'
  }
}

# node default {}       # uncomment this line if it doesn't already exist in your manifest
```

ceci implique, que la prochaine fois que puppet fera une mise à jour de host2 (agent node), ceci installera Apache package et configurera un hôte virtuel “exemple.com” il sera à l’écoute du port 80 et avec un document dans /var/www/html

au niveau de l'agent node

```
[root@node ~]# /opt/puppetlabs/bin/puppet agent --test<\code>
```

```
Il possible de gerer des modules et des manifestes en fontion de l'environnement de travail. PAr exemple, nous
pouvons considérer que le système gère 2 environnements : un environnement de production et un environnement de
développement.
<code>[root@puppet ~]# cd /etc/puppet
[root@puppet puppet]# ls
auth.conf  environments  fileserver.conf  manifests  modules  puppet.conf
[root@puppet puppet]# cd environments/
[root@puppet environments]# ls
example_env<\code>

<code>
[root@puppet environments]# mkdir -p production/manifests
[root@puppet environments]# mkdir -p production/modules
[root@puppet environments]# mkdir -p developpement/manifests
[root@puppet environments]# mkdir -p developpement/modules
```

6. Reférences

http://www.mit.edu/people/marthag/talks/puppet/img2.html
https://doc.ubuntu-fr.org/puppet
https://www.digitalocean.com/community/tutorials/how-to-install-puppet-4-in-a-master-agent-setup-on-centos-7
https://www.youtube.com/watch?v=61X2Armexf8
https://www.youtube.com/watch?v=k51SY_o9hMo
https://wiki.deimos.fr/Puppet_:_Solution_de_gestion_de_fichier_de_configuration
https://docs.puppet.com/puppet/4.6/reference/architecture.html