Juin 2023

Procédure d'installation des services NTP, SSH et Apache avec Puppet
Gestion des environnements Puppet avec Git

TP noté (notation indiquée ci-dessous)
Durée : 8 heures

On se propose d'écrire les *manifests* Puppet permettant de configurer automatiquement l'installation des services NTP, SSH et Apache, en fonction d'un certain nombre de paramètres propres au système, et dont la valeur sera donnée par *facter*.

Le code sera développé, et maintenu dans un dépôt Git dédié, dont la structure est donnée ci-dessous. Un *README.md* explicatif (qui tiendra lieu de court compte-rendu) sera écrit en anglais, au format Markdown.

Il est possible de s'aider de ressources trouvées sur internet, ce qui inclut chatGPT. Cependant, **tout ce qui est demandé a été vu en cours**. On ne demande donc pas de copier-coller des bouts de code trouvés sur internet, mais un code simple, repris des cours. De même, on peut s'aider, mais le plagiat sera sanctionné.

Partie 1 : GitLab / GitHub & Git

Le code sera déposé et accessible préférentiellement via le GitLab de Lyon 1 (vous me partagerez le lien vers votre dépôt, après m'avoir ajouté comme *developer* au dépôt) : *https://forge.univ-lyon1.fr/*

PS : vous pouvez aussi m'envoyer le dépôt Git par e-mail, une fois compressé (éviter le format zip).

Deux branches (au minimum) seront présentes dans le dépôt : les branches *develop* et *main.* Cette dernière sera considéré comme la branche contenant le code dit de production. Cette branche contiendra une version du code « prête à l'emploi », identifiée par le *tag* **1.0**. Ce code sera testé et devrait permettre l'installation des trois services NTP, SSH et Apache.

Note : vous pouvez aussi créer des branches spécifiques NTP, SSH et Apache, mais la branche main devra contenir, en fin de TP, tout le code requis pour l'installation de ces services, dans une version parfaitement fonctionnelle.

La bonne utilisation de Git sera prise en compte dans la note.

Rappels des bonnes pratiques d'utilisation de Git

- « commiter » régulièrement
- message clair pour chaque commit (en anglais de préférence pour les dépôts partagés)
  - si possible en une ligne avec l'option *-m (ou -am)* de la commande *git commit*
  - exemples :
    - Implementing feature machin-truc
    - Cleaning and refactoring of the class toto
    - bugfix #meltdown

Concernant la notation : 6 (**+1**) points sur la partie Git

- 2 points sur la bonne structure : branches et tags
- 2 points pour des messages clairs aux commits
- 2 points pour l'écriture du *README.md* en anglais, au format Markdown [*]
- **Bonus :** 1 point si utilisation de GitLab ou GitHub

[*] : voir, pour visualiser la syntaxe de ce format, http://markdownlivepreview.com/

Partie 2 : écriture des manifests et modules Puppet (4 heures)

Il s'agit donc de produire trois modules Puppet qui permettront d'installer un serveur SSH et les services NTP et Apache sous Ubuntu et CentOS. Ces modules pourront donc être utilisés sous les deux systèmes (prévoir un test conditionnel qui retourne une erreur si le système d'exploitation n'est pas compatible).

Dans le répertoire des modules, on trouvera les 3 répertoires suivants : *manifest*, *files* et *templates.* Le répertoire *Manifests* contiendra les codes Puppet, *files* les fichiers de configuration, et *templates* les fichiers de configuration contenant des variables.

Le répertoire *manifests contiendra les 5 fichiers suivants :*

- *init.pp* (gestion des dépendances des autres classes)
- *params.pp* (paramètres pertinents pour l'installation des paquets requis)
- *install.pp* (installation des paquets requis)
- *config.pp* (gestion des fichiers de configuration)
- *service.pp* (gestion du service)

Le code pourra être documenté, mais la façon d'utiliser ce code sera explicitée dans le *README.md* (expliciter toutes les commandes requises pour utiliser ses modules).

Concernant les expressions conditionnelles, on prévoira toujours une condition dite « par défaut ».

On utilisera les attributs *before* ou *subscribe* pour s'assurer que les ressources soient appliquées dans le bon ordre.

Concernant le module SSH, on gérera un fichier *template* « *ssh/templates/sshd_config.erb »* pour la configuration du service

- on fixera le protocole de connexion à la version **2 ;**
- on se connectera sur le port **23** si on est sur une machine physique, port **24** sur une machine virtuelle.

Concernant le service Apache, on configurera aussi le service via un fichier *template « apache/templates/000-default.conf.erb »* de façon à ce que :

- on fixera le port d'écoute au port 84 (au lien du port 80 par défaut) ;
- on fixera la racine du serveur web à un chemin spécifique qui permettra d'afficher une courte synthèse de ce projet.

Rappels concernant Puppet

Quelques commandes utiles :

- puppet module generate m*y_awesome_sshd*
- *sudo puppet apply --modulepath=/home/sebastien/Puppet/ -e "include my_awesome_sshd"*

La syntaxe du code Puppet peut être vérifiée avec les commandes suivantes :

- *puppet-lint*
- *puppet parser validate*

Une variable, dans un fichier *template*, est définie comme suit :
Protocol <%= @sshd_protocol %>

Une classe Puppet est définie comme suit :

```
class my_awesome_sshd::service {
        [...]
}
```

Le fichier *init.pp* aura une syntaxe de la forme suivante (un module = une classe pour Puppet) :

```
class my_awesome_sshd {
 # define dependencies between classes (not mandatory)
Class['my_awesome_sshd::params']~>Class['my_awesome_sshd::config']~>Class['my_awesome_
sshd::install']~>Class['my_awesome_sshd::service']
# include the required dependancies
include my_awesome_sshd::params, my_awesome_sshd::config, my_awesome_sshd::install,
my_awesome_sshd::service
}
```

On pourra créer un fichier *node.pp* pour appliquer les trois modules automatiquement, ce fichier ressemblera à :

```
node 'my_awesome_computer' {
  $sshd_port = 27
  $sshd_protocol = 2
  include my_awesome_sshd
  include my_awesome_ntp
}
```

Concernant la notation 8 (+**1**) points

- 2 points pour NTP
- 3 points pour SSH
  - 2 points pour les manifestes (si fonctionnels)
  - 1 point pour le fichier *template* (si fonctionnel)
- **Bonus** : 1 point si fichier *node.pp* fonctionnel (avec explication dans *README.md*)

Partie 3 : Les environnements Puppet (2 heures)

Git et Puppet sont traditionnellement utilisés conjointement dans le processus de développement, de test et de déploiement (mise en production) de recettes Puppet. Une branche dans Git correspondant à ce que l'on appelle un « environnement » dans Puppet. L'environnement par défaut est l'environnement dit de production, mais on utilise aussi des environnements de développement et/ou de test.

L'environnement par défaut est défini dans le fichier de configuration de Puppet, « */etc/puppet/puppet.conf » :*

[main]
modulepath = /chemin/vers/le/repertoire/modules
environment = production

Par la suite, on utilisera la variable d'environnement PUPPET_ENVIRONMENT pour définir l'environnement en cours :

PUPPET_ENVIRONMENT=test puppet apply --modulepath=/chemin/vers/le/depot-git/modules site.pp

Dans cet exemple, */chemin/vers/le/depot-git* représente le chemin local du dépôt Git cloné, et *modules* est le répertoire contenant les modules Puppet. Assurez-vous que les fichiers de configuration et les manifestes nécessaires, tels que *site.pp* (exemple ci-dessus), sont présents dans le répertoire approprié de la branche Git spécifiée. Par exemple, si vous utilisez la branche Git "test", le fichier *site.pp* devrait être présent dans le répertoire correspondant à cette branche.

À réaliser

Modifier le dépôt Git en ajoutant de nouvelles branches, elles correspondront aux environnements Puppet « production », « développement » et « test ».

La branche « développement » contient les codes en cours d'écriture, la branche « test » les codes en cours de test, et lorsque ceux-ci sont validés, la branche sera alors fusionnée sur « production ». On retrouvera donc cette structure dans le graphe de développement fourni par Git. La version de production (sur la branche de production) contenant les modules validés des 3 services sera alors taggués avec le nom **1.0.**

Créer maintenant une branche web (qui correspond donc à un environnement Puppet Web, destiné aux machines hébergeant des sites Web). Cette branche ne contient que le module Apache. Modifier maintenant ce module pour qu'une fois déployé, le site web affiche les informations suivantes : *uptime* de la machine, l'heure, le système d'exploitation, et sa version.

À quelle adresse peut-on accéder à ce site ?

Concernant la notation 6 points

- 3 points pour les 3 environnements Puppet (si commandes fournies et modules fonctionnels)
- 1 point si tag bien placé
- 2 points pour l'environnement web
  - 1 point pour le module mis-à-jour
  - 1 point si la page web affiche les informations demandées.