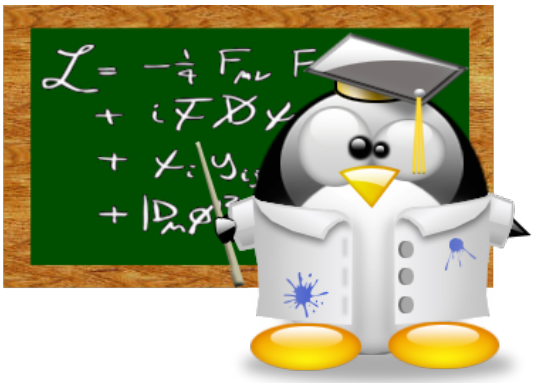

FORMATUX - Support de cours GNU/Linux

Automatisation - DevOPS

1.0, 24/06/2017

Chapitre 1. Préface

GNU/Linux est un **système d'exploitation** libre fonctionnant sur la base d'un **noyau Linux**, également appelé **kernel Linux**.

Linux est une implémentation libre du système **UNIX** et respecte les spécifications **POSIX**.

GNU/Linux est généralement distribué dans un ensemble cohérent de logiciels, assemblés autour du noyau Linux et prêt à être installé. Cet ensemble porte le nom de "**Distribution**".

- La plus ancienne des distributions est la distribution **Slackware**.
- Les plus connues et utilisées sont les distributions **Debian**, **Redhat** et **Arch**, et servent de base pour d'autres distributions comme **Ubuntu**, **CentOS**, **Fedora**, **Mageia** ou **Manjaro**.

Chaque distribution présente des particularités et peut être développée pour répondre à des besoins très précis :

- services d'infrastructure ;
- pare-feu ;
- serveur multimédia ;
- serveur de stockage ;
- etc.

La distribution présentée dans ces pages est la CentOS, qui est le pendant gratuit de la distribution RedHat. La distribution CentOS est particulièrement adaptée pour un usage sur des serveurs d'entreprises.

1.1. Crédits

Ce support de cours a été rédigé par les formateurs :

- Patrick Finet ;
- Antoine Le Morvan ;
- Xavier Sauvignon ;

1.2. Licence

Formatux propose des supports de cours Linux libres de droits à destination des formateurs ou des personnes désireuses d'apprendre à administrer un système Linux en autodidacte.

BY : Paternité. Vous devez citer le nom de l'auteur original.

SA : Partage des Conditions Initiales à l'Identique.

- Licence Creative Commons-BY-SA : https://creativecommons.org/licenses/by-sa/3.0/fr/
- Licence Art Libre : http://artlibre.org/

Les documents de formatux et leurs sources sont librements téléchargeables sur framagit :

- https://framagit.org/alemorvan/formatux.fr-support/

Vous y trouverez la dernière version de ce document.

A partir des sources, vous pouvez générer votre support de formation personnalisé. Nous vous recommandons le logiciel AscidocFX téléchargeable ici : http://asciidocfx.com/

1.3. Gestion des versions

Table 1. Historique des versions du document

Version	Date	Observations
1.0	Avril 2017	Version initiale.

Chapitre 2. Gestion de configurations avec Puppet

Il est difficile de s'appuyer sur des processus manuels ou des scripts personnalisés pour accomplir des tâches répétitives.

Lorsque les environnements grossissent ou que les équipes accueillent de plus en plus de techniciens, ces méthodes sont difficiles à maintenir et à optimiser. Elles peuvent causer des risques pour la sécurité du système, incluant des erreurs de configuration, ce qui au final, peut faire réduire la productivité.

La gestion automatique des configurations élimine beaucoup de travail manuel et augmente la réactivité des équipes. Cette réactivité devient de plus en plus importante avec la montée en puissance de la virtualisation, qui révolutionne notre gestion du cycle de vie des serveurs : durée de vie plus courte, déploiement plus rapide, configurations plus standardisées et conformes.

Parmi les systèmes de gestion de configurations, plusieurs systèmes ont fait leur apparition :

- puppet ;
- chef ;
- ansible ;
- etc.

Des outils ont également fait leur apparition pour encore faciliter l'administration de ces systèmes :

- geppetto : un environnement de développement (IDE) pour puppet ;
- the foreman : un gestionnaire de cycle de vie complet des serveurs.

2.1. La gestion de configuration

La gestion de configuration est le processus de standardisation des configurations de ressources et l'assurance de leur état dans l'infrastructure informatique, avec des méthodes automatisées mais agiles. Le management de configurations est devenu critique pour le succès des projets informatiques.

Concrètement, lors de l'ajout d'un nouveau serveur au sein d'une infrastructure informatique complexe, les administrateurs système ne doivent pas perdre de temps pour la configuration des briques de base du système : la configuration des services NTP, DNS, SMTP, SSH, la création des comptes utilisateurs, etc… doit être totalement automatisée et transparente aux équipes.

L'utilisation d'un gestionnaire de configuration doit également permettre d'installer un clone de serveur d'une manière totalement automatisée, ce qui peut être pratique pour des environnements multiples (Développement → Intégration → Pré-production → Production).

La combinaison d'outils de gestion de configuration avec l'utilisation de dépôts de gestion de versions, comme « git », permet de conserver un historique des modifications apportées au système.

2.2. Puppet

Puppet a été conçu pour fonctionner en mode client-serveur. Son utilisation en mode de fonctionnement « autonome » est également possible et facile. La migration vers un système de clients « Puppet / Master Puppet » n'est pas d'une réalisation complexe.

Puppet est un logiciel d'automatisation qui rend simple pour l'administrateur système le provisionnement (la description matérielle d'une machine virtuelle), la configuration et la gestion de l'infrastructure tout au long de son cycle de vie. Il permet de décrire l'état de configuration d'un ensemble hétérogène de stations de travail ou de serveurs et de s'assurer que l'état réel des machines correspond bien à l'état demandé.

Par sa structure de langage, il fait le lien entre les bonnes pratiques, le cahier de procédures et l'état effectif des machines.

Vocabulaire Puppet

- **Noeud** (Node) : serveur ou poste de travail administré par Puppet ;
- **Site** : ensemble des noeuds gérés par le Puppet Master ;
- **Classe** : moyen dans Puppet de séparer des morceaux de code ;
- **Module** : unité de code Puppet qui est réutilisable et pouvant être partagé ;
- **Catalogue** : ensemble des classes de configuration à appliquer à un nœud ;
- **Facter** : librairie multiplateforme qui fournit à Puppet sous forme de variables les informations propres au système (nom, adresse ip, système d'exploitation, etc.) ;
- **Ressource** (Resource): objet que Puppet peut manipuler (fichier, utilisateur, service, package, etc.) ;
- **Manifeste** (Manifest) : regroupe un ensemble de ressource.

Architecture

Puppet conseille de coupler son fonctionnement avec un gestionnaire de version type « git ».

Un serveur PuppetMaster contient la configuration commune et les points de différence entre machines ;

Chaque client fait fonctionner puppetd qui :

- applique la configuration initiale pour le nœud concerné ;
- applique les nouveautés de configuration au fil du temps ;

- s'assure de manière régulière que la machine correspond bien à la configuration voulue.

La communication est assurée via des canaux chiffrés, en utilisant le protocole https et donc TLS (une mini-pki est fournie).

Toute la configuration (le référentiel) de Puppet est centralisée dans l'arborescence /etc/puppet du serveur de référence :

- **/etc/puppet/manifests/site.pp** : est le premier fichier analysé par Puppet pour définir son référentiel. Il permet de définir les variables globales et d'importer des modules ;
- **/etc/puppet/manifests/node.pp** : permet de définir les nœuds du réseau. Un nœud doit être défini par le nom FQDN de la machine ;
- **/etc/puppet/modules/<module>** : sous-répertoire contenant un module.

2.3. Installation

Les dépôts Puppets doivent être activés :

```
[root]# vim /etc/yum/yum.repos.d/puppetlabs.repo
[puppetlabs-products]
name=Puppet Labs Products El 6 - $basearch
baseurl=http://yum.puppetlabs.com/el/6/products/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
enabled=1
gpgcheck=1

[puppetlabs-deps]
name=Puppet Labs Dependencies EL 6 - $basearch
baseurl=http://yum.puppetlabs.com/el/6/dependencies/$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
enabled=1
gpgcheck=1
```

puis :

```
[root]# yum update
[root]# yum install puppet
```

2.4. Hello world

Pour fonctionner, le client autonome puppet a besoin d'un fichier appelé manifeste, dont l'extension sera en « .pp ».

Le langage utilisé est le ruby.

Créer un fichier /root/config-init.pp :

```
[root]# vim /root/config-init.pp
file {'helloworld':
  path => '/tmp/helloworld',
  ensure => present,
  mode => 0640,
  content => "Helloworld via puppet ! "
}
```

Exécuter ce manifeste avec la commande puppet :

```
[root]# puppet apply /root/config-init.pp
Notice : Compiled catalog for centos6 in environnement production in 0.31 seconds
Notice: /Stage[main]/main/File[helloworld]/ensure: created
Notice: Finished catalog run in 0.07 seconds
```

2.5. Les modules Puppet

Les modules complémentaires à Puppet peuvent être téléchargés sur le site puppetlabs.

L'installation d'un module complémentaire pourra se faire, soit directement depuis internet, soit par le téléchargement manuel de l'archive .tar.gz.

Depuis internet :

```
puppet module install nomdumodule
```

Une commande de recherche de modules existe :

```
puppet module search nomdumodule
```

Pour rechercher les modules installés :

```
puppet module list
```

Et pour les mettre à jour :

```
puppet module upgrade nomdumodule
```

Pensez à préciser le proxy dans la commande puppet en exportant les variables http_proxy et https_proxy :

```
export http_proxy=http://10.10.10.7:8080
export https_proxy=http://10.10.10.7:8080
```

Sans connexion internet

Sans connexion internet, un module peut être installé en fournissant dans la commande puppet le chemin vers le fichier tar.gz :

```
puppet module install ~/nomdumodule.tar.gz --ignore-dependencies
```

Grâce aux modules du dépôt Puppet, les possibilités de l'outil sont quasiment infinies. Le téléchargement et l'utilisation des modules du dépôt permettent un gain de temps confortable car l'outil nécessite peu de compétences en développement.

2.6. Documentation

La liste des types et leurs attributs est consultable en ligne :

- https://docs.puppetlabs.com/references/latest/type.html

Une documentation de formation est également consultable :

- https://doc.puppetlabs.com/learning/introduction.html

2.7. Commandes utiles

Lister les objets connus par puppet :

```
puppet describe -l
```

Lister les valeurs possibles d'une ressource :

```
puppet describe user
```

Lister les objets du système :

```
puppet resource user
```

2.8. Cas concrets

La création d'un noeud (node)

Le code du manifeste doit être découpé en classe pour des raisons de maintenance et d'évolutivité.

Un objet de type node recevra les classes à exécuter. Le node « default » est automatiquement exécuté.

Le fichier site.pp contiendra l'ensemble du code :

```
node default {
  include init_services
}
```

Gestion des services

La classe init_services contient les services qui doivent être lancés sur le nœud et ceux qui doivent être stoppés :

```
class init_services {

  service { ["sshd","NetworkManager","iptables","postfix","puppet","rsyslog","sssd",
"vmware-tools"]:
    ensure => 'running',
    enable => 'true',
  }

  service { ["atd","cups","bluetooth","ip6tables","ntpd","ntpdate","snmpd","snmptradp"]:
    ensure => 'stopped',
    enable => 'false',
  }
}
```

La ligne ensure ⇒ a pour effet de démarrer ou non un service, tandis que la ligne enable ⇒ activera ou non le démarrage du service au démarrage du serveur.

Gestion des utilisateurs

La classe create_users contiendra les utilisateurs de notre système. Pensez à ajouter l'appel de cette classe dans le node default !

```
class create_users {
  user { 'antoine':
    ensure => present,
    uid => '5000',
    gid => 'users',
    shell => '/bin/bash',
    home => '/home/antoine',
    managehome => true,
  }
}
```

```
node default {
  include init_services
  include create_users
}
```

Au départ du personnel pour mutation, il sera aisé de supprimer son compte en remplaçant la ligne ensure ⇒ present par ensure ⇒ absent et en supprimant le reste des options.

La directive managehome permet de créer les répertoires personnels à la création des comptes.

La création des groupes est toute aussi aisée :

```
group { "DSI":
  ensure => present,
  gid => 1001
}
```

Gestion des dossiers et des fichiers

Un dossier peut être créé avec la ressource « file » :

```
file { '/etc/skel/boulot':
  ensure => directory,
  mode   => 0644,
}
```

Un fichier peut être copié d'un répertoire vers un autre :

```
file { '/STAGE/utilisateurs/gshadow':
  mode   => 440,
  owner  => root,
  group  => root,
  source => "/etc/gshadow"
}
```

Modification de valeurs

La commande augeas, développée par la société RedHat, permet de modifier les valeurs des variables dans les fichiers de configuration. Son utilisation peut s'avérer autant puissante que complexe.

Un usage minimaliste serait par exemple :

```
augeas { "Modification default login defs" :
    context => "/files/etc/login.defs",
    changes => ["set UID_MIN 2000","set GID_MIN 700","set PASS_MAX_DAYS 60"],
}
```

Le contexte est suffixé de « /files/ » pour préciser qu'il s'agit d'un système de fichiers local.

Exécution d'une commande externe

La commande exec est utilisée pour lancer une commande externe :

```
exec { "Redirection":
  command => "/usr/sbin/useradd -D > /STAGE/utilisateurs/defaut",
}
```

De manière générale, les commandes et les fichiers doivent être décrits en absolu dans les manifestes.

Rappel : la commande whereis fournit le chemin absolu d'une commande.

Chapitre 3. Ansible

Ansible centralise et automatise les tâches d'administration. Il est sans agent (il ne nécessite pas de déploiements spécifiques sur les clients) et utilise le protocole SSH pour configurer à distance les clients Linux.

L'interface graphique web d'Ansible est payante.

L'ouverture des flux SSH vers l'ensemble des clients depuis le serveur Ansible font de lui un elément critique de l'architecture qu'il faudra attentivement surveiller.

Pourquoi scripter en Bash n'est pas considéré comme de l'automatisation ? (langage Impératif vs Déclaratif)

Même si la connaissance du Bash est une exigence de base pour un administrateur système, celui-ci est un langage de programmation interprété "**impératif**". Il exécute les instrutions les unes à la suite des autres.

Les langages dédiés au domaine (DSL Domain Specific Language) comme Ansible ou Puppet, quant à eux, ne spécifient pas les étapes à réaliser mais l'état à obtenir.

Parce qu'Ansible est un langage déclaratif, il est très simple. Il suffit de lui dire "Fais cette action" ou "Met le serveur dans cet état". Ansible considèrera l'état désirée indépendamment de l'état initial ou du contexte. Peu importe dans quel état le serveur se situe au départ, les étapes à franchir pour arriver au résultat, le serveur est mis dans l'état désiré avec un rapport de succès (avec ou sans changement d'état) ou d'échec.

La même tâche d'automatisation en bash nécessiterait de vérifier tous les états possibles, les autorisations, etc. afin de déterminer la tâche à accomplir avant de lancer l'exécution de la commande, souvent en imbriquant de nombreux "si", ce qui complique la tâche, l'écriture et la maintenance du script.

3.1. Installation sur le serveur

Ansible est disponible dans le dépot EPEL :

- Installation d'EPEL :

```
$ sudo yum install epel-release
```

La configuration du serveur se situe sous /etc/ansible.

Deux fichiers de configuration :

- Le fichier de configuration principal *ansible.cfg* : commandes, modules, plugins, configuration ssh ;
- Le fichier de gestion des machines clientes *hosts* : déclaration des clients, des groupes.

Le fichier /etc/ansible/hosts

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com
```

3.2. La commande ansible

La commande ansible lance une tâche sur un ou plusieurs hôtes cibles.

Syntaxe de la commande ansible

```
ansible <host-pattern> [-m module_name] [-a args] [options]
```

Exemples :

- Lister les hôtes appartenant à un groupe :

```
ansible {{group}} --list-hosts
```

- Pinger un groupe d'hôtes avec le module ping :

```
ansible {{group}} -m ping
```

- Afficher des facts d'un groupe d'hôtes avec le module setup :

```
ansible {{group}} -m setup
```

- Exécuter une commande sur un groupe d'hôtes en invoquant le module command avec des arguments :

```
ansible {{group}} -m command -a '{{commande}}'
```

- Exécuter une commande avec les privilèges d'administrateur :

```
ansible {{group}} --become --ask-become-pass -m command -a '{{commande}}'
```

- Exécuter une commande en utilisant un fichier d'inventaire personnalisé :

```
ansible {{group}} -i {{inventory_file}} -m command -a '{{commande}}'
```

Table 2. Options principales de la commande ansible

Option	Information
-a 'arguments'	Les arguments à passer au module.

Option	Information
-b -K	Demande un mot de passe et lance la commande avec des privilèges supérieurs.
--user=utilisateur	Utilise cet utilisateur pour se connecter à l'hôte cible au lieu d'utiliser l'utilisateur courant.
--become -user=utilisateur	Exécute l'opération en tant que cet utilisateur (defaut : root).
-C	Simulation. Ne fait pas de changement sur la cible mais la teste pour voir ce qui devrait être changé.
-m module	Exécute le module appelé

3.3. Gestion des clients

Les serveurs clients doivent être ajoutés dans le fichiers /etc/ansibles/hosts.

Un groupe "centos6" est créé :

Le fichier /etc/ansible/hosts

```
[centos6]
172.16.1.217
172.16.1.192
```

Tester avec le module ping

Par défaut la connexion par mot de passe n'est pas autorisée par Ansible.

Décommenter la ligne suivante de la section [defaults] dans le fichier de configuration /etc/ansible/ansible.cfg :

```
ask_pass      = True
```

Lancer un ping sur chacun des serveurs du groupe CentOS 6 :

```
# ansible centos6 -m ping
SSH password:
172.16.1.192 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.16.1.217 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Le mot de passe root des serveurs distants vous est demandé, ce qui pose un problème de sécurité...

Authentification par clef

L'authentification par mot de passe va être remplacée par une authentification par clefs privée/publique beaucoup plus sécurisée.

Création d'une clef SSH

La bi-clefs va être générée avec la commande **ssh-keygen** :

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RpYuJzkkaeZzve8La8Xd/8kTTE8t43DeS+L7WB26WF8 root@ansible-srv
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|     .   .       |
|    = . +       .|
|   + o *    . +.o|
|    o * S. . *o*.|
|     o * .o ..=+=|
|        o.  .oooE|
|        .+  o.*o+|
|       ...+o +o=+|
+----[SHA256]-----+
```

La clef publique peut être copiée sur les serveurs :

```
# ssh-copy-id root@172.16.1.192
# ssh-copy-id root@172.16.1.217
```

Re-commenter la ligne suivante de la section [defaults] dans le fichier de configuration /etc/ansible/ansible.cfg pour empêcher l'authentification par mot de passe :

```
#ask_pass      = True
```

Test d'authentifcation par clef privée

Pour le prochain test, le module shell, permettant l'exécution de commandes à distance, est utilisé :

```
# ansible centos6 -m shell -a "uptime"
172.16.1.192 | SUCCESS | rc=0 >>
 12:36:18 up 57 min,  1 user,  load average: 0.00, 0.00, 0.00

172.16.1.217 | SUCCESS | rc=0 >>
 12:37:07 up 57 min,  1 user,  load average: 0.00, 0.00, 0.00
```

Aucun mot de passe n'est demandé, l'authentification par clef privée/publique fonctionne !

Exemple de connexion à une instance Cloud Amazon ECS

Lors de la création d'une instance Amazon, une clef privée est créée et téléchargée sur le poste local.

Ajout de la clef dans l'agent SSH :

```
ssh-add path/to/fichier.pem
```

Lancement des facts sur les serveurs aws :

```
ansible aws --user=ec2-user --become -m setup
```

3.4. Utilisation

Ansible peut être utilisé depuis l'interpréteur de commandes ou via des playbooks.

Les modules

La liste des modules classés par catégories se trouve à l'adresse http://docs.ansible.com/ansible/modules_by_category.html. Ansible en propose plus de 750 !

Un module s'invoque avec l'option -m de la commande ansible

Exemples d'installation logiciel

Le module yum permet d'installer des logiciels sur les clients cibles :

```
# ansible centos6 -m yum -a name="httpd"
172.16.1.192 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
      ...
      \n\nComplete!\n"
    ]
}
172.16.1.217 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
      ...
    \n\nComplete!\n"
    ]
}
```

Le logiciel installé étant un service, il faut maintenant le démarrer avec le module service (centos 6) ou systemd (centos 7) :

```
# ansible centos6 -m service -a "name=httpd state=started"
172.16.1.192 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
172.16.1.217 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
```

Les playbooks

Les playbooks ansible décrivent une politique à appliquer à des systèmes distants, pour forcer leur configuration. Les playbooks sont écris dans un format texte facilement compréhensible regroupant un ensemble de tâches : le format yaml.

En savoir plus sur le yaml : http://docs.ansible.com/ansible/YAMLSyntax.html

Syntaxe de la commande ansible-playbook

```
ansible-playbook <fichier.yml> ... [options]
```

Les options sont identiques à la commande ansible.

La commande renvoi les codes d'erreurs suivants :

Table 3. Codes de sortie de la commande ansible-playbook

0	**OK ou aucun hôte correspondant**
1	Erreur
2	Un ou plusieurs hôtes sont en échecs
3	Un ou plusieurs hôtes ne sont pas joignables
4	Erreur d'analyse
5	Mauvaises options ou options incomplètes
99	Execution interrompue par l'utilisateur
250	Erreur inattendue

Exemple de playbook apache et mysql

Le playbook suivant permet d'installer apache et mysql sur nos serveurs cibles :

```
---
- hosts: centos6
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd,php,php-mysqli state=latest
  - name: ensure httpd is started
    service: name=httpd state=started
  - name: ensure mysql is at the latest version
    yum: name=mysql-server state=latest
  - name: ensure mysqld is started
    service: name=mysqld state=started
```

L'exécution du playbook s'effectue avec la commande **ansible-playbook** :

```
$ ansible-playbook test

PLAY [centos6] ****************************************************************

TASK [setup] ******************************************************************
ok: [172.16.1.192]
ok: [172.16.1.217]

TASK [ensure apache is at the latest version] *********************************
ok: [172.16.1.192]
ok: [172.16.1.217]

TASK [ensure httpd is started] ************************************************
changed: [172.16.1.192]
changed: [172.16.1.217]

TASK [ensure mysql is at the latest version] **********************************
changed: [172.16.1.192]
changed: [172.16.1.217]

TASK [ensure mysqld is started] ***********************************************
changed: [172.16.1.192]
changed: [172.16.1.217]

PLAY RECAP ********************************************************************
172.16.1.192               : ok=5    changed=3    unreachable=0    failed=0
172.16.1.217               : ok=5    changed=3    unreachable=0    failed=0
```

Exemple de préparation d'un noeud mysql

Dans ce playbook, deux serveurs Cloud vont être préparés pour devenir des serveurs Multi-Maître MySQL (voir cours MySQL Multi-Maître).

Le playbook utilise :

- Des variables ;
- Ajoute des lignes dans le fichier /etc/hosts ;
- Installe et démarre MariaDB ;
- Créer une base de données, un utilisateur et lui donne tous les droits sur les bases de données.

```
---

- hosts: aws
  remote_user: ec2-user
  become: yes
  vars:
    mysqlpackage: "mariadb-server,MySQL-python"
    mysqlservice: "mariadb"
    mysql_port: "3306"
    dbuser: "synchro"
    dbname: "mabase"
    upassword: "M!rro!r"

  tasks:
    - name: configurer le noeud 1 dans /etc/hosts
      lineinfile:
              dest: /etc/hosts
              line: "13.59.197.48 mirroir1.local.lan mirroir1"
              state: present
    - name: configurer le noeud 2 dans /etc/hosts
      lineinfile:
              dest: /etc/hosts
              line: "52.14.125.109 mirroir2.local.lan mirroir2"
              state: present
    - name: mariadb installe et a jour
      yum: name="{{ mysqlpackage }}" state=latest
    - name: mariadb est demarre
      service: name="{{ mysqlservice }}" state=started
    - name: creer la base de donnee
      mysql_db: name="{{ dbname }}" state=present
    - name: creer un utilisateur
      mysql_user: name="{{ dbuser }}" password="{{ upassword }}" priv=*.*:ALL host='%'
state=present
    - name: restart mariadb
      service: name="{{ mysqlservice }}" state=restarted
...
```

Chapitre 4. Glossaire

BASH
: Bourne Again SHell

BIOS
: Basic Input Output System

CIDR
: Classless Inter-Domain Routing

Daemon
: Disk And Execution MONitor

DHCP
: Dynamic Host Control Protocol

DNS
: Domain Name Service

FQDN
: Fully Qualified Domain Name

LAN
: Local Area Network

NTP
: Network Time Protocol

nsswitch
: Name Service Switch

POSIX
: Portable Operating System Interface

POST
: Power On Self Test

SHELL
: En français "coquille". Á traduire par "interface système".

SMB
: Server Message Block

SMTP

Simple Mail Tranfer Protocol

SSH

Secure SHell

TLS

Transport Layer Security, un protocole de cryptographie pour sécuriser les communications IP.

TTY

teletypewriter, qui se traduit téléscripteur. C'est la console physique.

UEFI

Unified Extensible Firmware Interface

Chapitre 5. Index