



puppet

Puppet V6

Wajih HAJJI © 2018

61755866ee
Global Knowledge

Sommaire

- Introduction
- Installation
- Création et application d'un Manifest Puppet
- Gestion des Paquets et des services
- Puppet et Git

Chapitre 1

Introduction

61755866ee
Global Knowledge

Système de gestion centralisée des configurations (SCM)

- Selon Wikipedia : un système de gestion des configuration est défini comme suit :

« La **gestion de configuration** consiste à gérer **la description technique** d'un système¹ (et de ses divers composants), ainsi qu'à **gérer** l'ensemble **des modifications apportées** au cours de l'évolution du système.

En d'autres termes, il s'agit de **l'ensemble des processus** permettant d'assurer la **conformité** d'un produit aux **exigences**, tout au long de son cycle de vie.

La gestion de configuration est utilisée pour la configuration de systèmes complexes »

https://fr.wikipedia.org/wiki/Gestion_de_configuration_logicielle

Système de gestion centralisée des configurations (SCM)

- Résoud les problèmes d'administration systèmes de grands parcs :
 - Taches répétitives
 - Supervision de l'état du parc
 - Audit
 - etc.

Système de gestion centralisée des configurations (SCM)

- Méthodes possibles d'administration de grands parcs
 - Manuellement → long et source d'erreurs humaines
 - Boucle SSH (script)
 - Long!
 - Comment vérifier les résultats ?
 - Reprise sur erreur
 - Pré-requis: s'assurer que tous les systèmes sont ISO (ou s'assurer que tous les cas de figure sont gérés)
 - Update.sh (script au boot ou par cron/scheduler)
 - comment s'assurer que tous les nœuds ont bien exécuté le script ?
 - mêmes problèmes
 - comment distinguer les nœuds ?
 - Agent
 - Puppet
 - Infrastruct (Ruby)

Systeme de gestion centralisée des configurations (SCM)

- Exemple de Système de gestion centralisée des configurations (SCM)
 - *Chef*
 - *CFEngine*
 - *Salt*
 - *Ansible*



Puppet

- Puppet est un outil permettant une administration centralisée de machines.
 - C'est à dire qu'il permet de centraliser sur une machine la configuration d'un parc de machines
- C'est un outil open source et écrit en **Ruby** qui fonctionne sur tous les systèmes d'exploitation Unix et qui a un fonctionnement de type client/serveur.
 - Plusieurs clients se connecteront à un serveur contenant leurs configurations respectives.
- **Puppet** fournit un langage permettant de spécifier la configuration de chaque système pour permettre de simplifier sa propre configuration.
- → Pour simplifier, Puppet est la couche abstraite entre l'administrateur système et ses systèmes. L'administrateur spécifie les configurations à apporter sur les systèmes et Puppet les applique.
- <https://www.youtube.com/watch?v=QFcqvBk1gNA>

Pourquoi faire ?

- Eviter les tâches répétitives : on ne fait le travail de configuration qu'une seule fois
- Permettre une configuration homogène de parc
- S'assurer que les machines conservent leur configuration
- Déploiement rapide de machine
- Sécurité
 - Plus facile de déployer massivement un durcissement
 - Écart par rapport à la cible → reporting et remise en état

Puppet

- Orienté administration système
 - Couche d'abstraction de ressources
 - services, paquets, fichiers, utilisateurs,cron
- Configuration puissante (tests conditionnels, dépendance, héritage, template, ...)
- Code compact et lisible
- L'indisponibilité du serveur n'est pas bloquante
- Rentrer/sortir dans le système Puppet = installer/supprimer le daemon client
- Extensible : on peut commencer par ne gérer que quelques paramètres/fichiers, puis accroître la surface progressivement.

Declaratif vs. imperatif Les concepts principaux

Comparison

Imperative Shell Code

```
if [ 0 -ne $(getent passwd elmo > /dev/null)$? ]
then
    useradd elmo --gid sysadmin -n
fi

GID=`getent passwd elmo | awk -F: '{print $4}'`
GROUP=`getent group $GID | awk -F: '{print $1}'`

if [ "$GROUP" != "$GID" ] && [ "$GROUP" != "sysadmin" ]
then
    usermod --gid $GROUP $USER
fi
```

```
if [ "`getent group sysadmin | awk -F: '{print $1}'`" == "" ]
then
    groupadd sysadmin
fi
```

Declarative Puppet Code

```
user { 'elmo':
  ensure => present,
  gid   => 'sysadmin',
}
```

```
group { 'sysadmin':
  ensure => present,
}
```

Exemple pratique : Configuration site Web

- Voyons ensemble les étapes pour la configuration d'un site web
 - Sans Configuration management
 - Avec Configuration Management

Exemple pratique : Configuration site Web

- Sans Configuration Management :

- Se loger sur le serveur web
 - Installer les packages
 - Changer la configuration de Apache
 - Restart Apache
- Se loger sur le serveur d'application :
 - Installation et déploiement
 - Changement du config
 - Restart de Apach
- Se loger sur le serveur de la base de données
 - Execution script SQL
 - Changemnt de config
 - Restart

→ Le site a eu beaucoup de succès

- Il faut installer 10 nœuds



Exemple pratique : Configuration site Web

- Définir la configuration du site web , de l'application et de la base de données dans un DSL (Domain Specific Language)
- Ecrire un test unitaire du code de la configuration et le tester
- Déployer la configuration Puppet vers les serveurs (web , Application et base de données)

→ Le site a eu beaucoup de succès

- Il faut installer 10 nœuds



→ Déployer le même code sur les 10 nœuds

Chapitre 2

Installation

61755866ee
Global Knowledge

Prérequis

- GIT
 - Si ce n'est pas déjà fait, commencez par créer votre compte sur GitHub <https://github.com/>
 - Si ce n'est pas déjà fait, naviguez à l'URL <https://git-scm.com/download> et suivez les instructions pour installer Git sur votre système d'exploitation
- VirtualBox
 - Si ce n'est pas déjà fait, naviguez à l'URL <https://www.virtualbox.org/wiki/Downloads> et suivez les instructions pour installer VirtualBox sur votre système d'exploitation.
- Vagrant
 - Si ce n'est pas déjà fait, naviguez à l'URL <https://www.vagrantup.com/downloads.html> et suivez les instructions pour installer Vagrant sur votre système d'exploitation.

Installation

- Clonez ensuite les fichiers pour ce cours :

Sous Debian/Ubuntu

```
hnorris@hnorris-E200HA:~$ git clone https://github.com/bitfield/puppet-beginners-guide-3.git
Clonage dans 'puppet-beginners-guide-3'...
remote: Counting objects: 854, done.
remote: Total 854 (delta 0), reused 0 (delta 0), pack-reused 854
Réception d'objets: 100% (854/854), 98.47 KiB | 320.00 KiB/s, fait.
Résolution des deltas: 100% (443/443), fait.
```

Sous Windows

```
C:\Users\hnorris>git clone https://github.com/bitfield/puppet-beginners-guide-3.git
Cloning into 'puppet-beginners-guide-3'...
remote: Counting objects: 854, done.
remote: Total 854 (delta 0), reused 0 (delta 0), pack-reused 854R
Receiving objects: 100% (854/854), 98.47 KiB | 202.00 KiB/s, done.
Resolving deltas: 100% (443/443), done.
```

Installation

- Installer la Machine Virtuelle Puppet

- Sous Debian/Ubuntu

```
hnorris@hnorris-E200HA:~$ cd puppet-beginners-guide-3  
hnorris@hnorris-E200HA:~/puppet-beginners-guide-3$ scripts/start_vagrant.sh
```

- Sous Windows

- Ajoutez le chemin **C:\Program Files\Git\bin** à votre PATH et au PATH système.
 - Exécutez **cmd** et saisissez les commandes suivantes :

```
C:\Users\hnorris\puppet-beginners-guide-3>cd scripts  
C:\Users\hnorris\puppet-beginners-guide-3\scripts>start_vagrant.sh
```

- **Git bash** sera lancé pour exécuter le script.

Installation

- Se connecter à la Machine Virtuelle Puppet
 - A l'issue du processus d'installation, vous obtiendrez

```
...
default: The `minitar` executable is no longer bundled with `minitar`. If you are
default: expecting this executable, make sure you also install `minitar-cli`.
default: Successfully installed minitar-0.6.1
default: Successfully installed puppet_forge-2.2.9
default: Successfully installed r10k-2.6.2
default: 10 gems installed
```

- Connectez-vous à la machine virtuelle en utilisant la commande **vagrant ssh** :

```
hnorris@hnorris-E200HA:~/puppet-beginners-guide-3$ vagrant ssh
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-127-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 Get cloud support with Ubuntu Advantage Cloud Guest:
   http://www.ubuntu.com/business/services/cloud

 0 packages can be updated.
```

```
C:\Users\hnorris\puppet-beginners-guide-3\scripts>vagrant ssh
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-127-generic x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
  http://www.ubuntu.com/business/services/cloud
```

```
0 packages can be updated.
0 updates are security updates.
```

```
vagrant@ubuntu-xenial:~$
```

Installation

- Vérifiez que Puppet a bien été installé
 - Ubuntu

```
vagrant@ubuntu-xenial:~$ puppet --version
5.5.1
vagrant@ubuntu-xenial:~$ exit
logout
Connection to 127.0.0.1 closed.
```

- Windows

```
vagrant@ubuntu-xenial:~$ puppet --version
5.5.1
vagrant@ubuntu-xenial:~$ exit
logout
Connection to 127.0.0.1 closed.

C:\Users\hnorris\puppet-beginners-guide-3\scripts>
```

Chapitre 3

Création et application d'un Manifest Puppet

61755866ee
Global Knowledge

C'est quoi un Manifest Puppet

- Puppet utilise son propre langage (DSL : Domain Specific Language)
- Puppet se spécialise dans le management des ressources (les fichiers , les utilisateurs ???)
- Puppet gère des ressources spécifiques
- Par défaut il n' y a pas d'ordre pour la gestion de ces ressources, il faut le faire explicitement
- Un manifest est un fichier de type (.pp) qui contient le langage de Puppet

C'est quoi un catalog

- Les manifest peuvent contenir des structures conditionnelles et itératives
- Puppet compile les manifestes en catalogue
- Un catalogue ne contient pas de structure conditionnelle et itérative
- Un catalogue contient uniquement les ressources et leur dépendances
- Un catalogue est valide uniquement pour un nœud
- Un nœud applique un catalogue



LAB #2 - Gestion des Fichiers

- Re-connectez-vous à votre machine virtuelle et créez le fichier `file_hello.pp`

```
vagrant@ubuntu-xenial:~$ vi file_hello.pp
vagrant@ubuntu-xenial:~$ cat file_hello.pp
file { '/tmp/hello.txt':
  ensure  => file,
  content => "hello, world\n",
}
```

- Dans ce fichier nous pouvons constater la déclaration d'une ressource de type fichier avec `file`. Une ressource est une configuration que vous souhaitez être gérée par Puppet par exemple,
 - un fichier, un utilisateur, un compte ou bien un paquet. Le format de notre manifest est donc

```
TYPE_RESSOURCE { TITRE:
  ATTRIBUT => VALEUR,
  ...
}
```



LAB #2 - Gestion des Fichiers

- La ressource est identifiée par un titre.
 - Chaque ressource doit avoir un titre unique. Dans le cas d'un fichier c'est le chemin complet vers le fichier - `/tmp/hello.txt`.
- Le reste du code est une liste d'attributs pour la ressource. Dans le cas d'un fichier les attributs disponibles pour sont, par exemple :
 - le contenu,
 - le propriétaire,
 - le groupe,
 - le mode,
 - etc ...
- → **Important** - Bien que les attributs soient différents selon le type de ressource, l'attribut `ensure` est commun à toutes les ressources. Par contre, la valeur de cet attribut diffère selon le type de ressource. Dans le cas de notre code, il est stipulé un fichier et non un répertoire ou un liens symbolique.



LAB #2 - Gestion des Fichiers

- Le contenu de ce fichier est stipulé par l'attribut **content** qui est ici une chaîne *hello, world* suivie par un caractère de nouvelle ligne *\n*.
 - L'application du manifest par Puppet se résume ainsi :
 - Puppet lit le manifest ainsi que la liste des ressources,
 - Puppet compile les ressources en un **catalogue**,
 - Puppet lit le catalogue et applique chaque ressource à tour de rôle.
- ➔ Le nom du manifest n'est pas important, par contre l'extention doit être **.pp**



LAB #2 - Gestion des Fichiers

- Compiler le fichier

```
vagrant@ubuntu-xenial:~$ sudo puppet apply file_hello.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.06 seconds
Notice: /Stage[main]/Main/File[/tmp/hello.txt]/ensure: defined content as '{md5}22c3683b094136c3398391ae71b20f04'
Notice: Applied catalog in 0.06 seconds
```

- Vérifiez que Puppet a écrit le fichier `/tmp/hello.txt` :

```
vagrant@ubuntu-xenial:~$ cat /tmp/hello.txt
hello, world
```



LAB #2 - Gestion des Fichiers

- *Modification d'un Fichier Existant sur le Serveur*
- Dans le cas où le fichier existe déjà et son contenu est différent, Puppet écrasera son contenu avec celui du manifest :

```
vagrant@ubuntu-xenial:~$ sudo sh -c 'echo "goodbye, world" > /tmp/hello.txt'
vagrant@ubuntu-xenial:~$ cat /tmp/hello.txt
goodbye, world
vagrant@ubuntu-xenial:~$ sudo puppet apply file_hello.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.06 seconds
Notice: /Stage[main]/Main/File[/tmp/hello.txt]/content: content changed '{md5}767887814e925822027f4fe63fb69ce2'
to '{md5}22c3683b094136c3398391ae71b20f04'
Notice: Applied catalog in 0.11 seconds
vagrant@ubuntu-xenial:~$ cat /tmp/hello.txt
hello, world
```



LAB #2 - Gestion des Fichiers

- *Effectuer un Dry Run avec Puppet*

- Il est possible de demander à Puppet, grâce à l'utilisation de l'option `--noop`, de nous informer des modifications qu'il aurait fait en appliquant un manifest, sans que ces modifications soient réellement effectuées :

```
vagrant@ubuntu-xenial:~$ sudo sh -c 'echo "goodbye, world" > /tmp/hello.txt'
vagrant@ubuntu-xenial:~$ sudo puppet apply --noop file_hello.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.07 seconds
Notice: /Stage[main]/Main/File[/tmp/hello.txt]/content: current_value '{md5}767887814e925822027f4fe63fb69ce2',
should be '{md5}22c3683b094136c3398391ae71b20f04' (noop)
Notice: Class[Main]: Would have triggered 'refresh' from 1 event
Notice: Stage[main]: Would have triggered 'refresh' from 1 event
Notice: Applied catalog in 0.07 seconds
vagrant@ubuntu-xenial:~$ cat /tmp/hello.txt
goodbye, world
```

- Comme vous pouvez constater, Puppet décide si un fichier doit être modifié en fonction de la valeur du hash md5.
 - Pour constater les modifications que Puppet aurait effectué, utilisez l'option `--show_diff` :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply --noop --show_diff file_hello.pp
```

Chapitre 4

Gestion des Paquets et des services

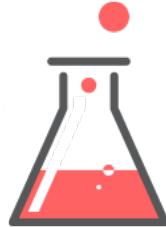
6175866ee
Global Knowledge

Utilisation des Paquets

- Utiliser la ressource de type package pour installer, supprimer ou mettre a jours des packages
- Support de plusieurs système de gestion de pckages (rpm, apt, aka ...)
- Les paramètres de la ressource package :
 - Name : titre de la ressource ou le nom du package a gérer
 - Ensure : installed, latest, held, exact, version , absent ...
 - Provider : le fournisseur a utiliser (exemple : gem)
 - Install_options :
- Exemple :

```
package { 'package_name':  
ensure => installed,  
}
```

Ce manifest aura comme résultat d'assurer que le package **package_name** soit installé en utilisant le gestionnaire des paquets du système d'exploitation.



LAB #3 - Gestion des Paquets

- Créez le fichier `package.pp` suivant :

```
vagrant@ubuntu-xenial:~$ vi package.pp
vagrant@ubuntu-xenial:~$ cat package.pp
package { 'cowsay':
  ensure => installed,
}
```

→ **Important** - Le titre de la ressource de type paquet est **cowsay**

- Appliquez le manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply package.pp
```

→ Le résultat de l'application de ce manifest est l'installation du paquet **cowsay**:



LAB #3 - Gestion des Paquets

```
vagrant@ubuntu-xenial:~$ dpkg --get-selections | grep cowsay
cowsay                      install
cowsay-off                  install
vagrant@ubuntu-xenial:~$ dpkg -s cowsay
Package: cowsay
Status: install ok installed
Priority: optional
Section: games
Installed-Size: 90
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: all
Version: 3.03+dfsg1-15
Depends: libtext-charwidth-perl, perl
Recommends: cowsay-off
Suggests: filters
Description: configurable talking cow
  Cowsay (or cowthink) will turn text into happy ASCII cows, with
  speech (or thought) balloons. If you don't like cows, ASCII art is
  available to replace it with some other creatures (Tux, the BSD
  daemon, dragons, and a plethora of animals, from a turkey to
  an elephant in a snake).
Original-Maintainer: Tony Mallefaud <maltouzes@gmail.com>
Homepage: https://web.archive.org/web/20120527202447/http://www.nog.net/~tony/warez/cowsay.shtml
```



LAB #3 - Gestion des Paquets

- Pour voir la version du paquet que Puppet pense être installé, utilisez la commande `puppet resource` en spécifiant le paquet concerné :

```
vagrant@ubuntu-xenial:~$ sudo puppet resource package openssl
package { 'openssl':
  ensure => '1.0.2g-1ubuntu4.12',
}
```

- L'utilisation de cette commande sans spécifier un paquet permet de voir la liste de tous les paquets :

```
vagrant@ubuntu-xenial:~$ sudo puppet resource package | more
```

```
package { 'acl':
  ensure => '2.2.52-3',
}
package { 'acpid':
  ensure => '1:2.0.26-1ubuntu2',
}
package { 'adduser':
  ensure => '3.113+nmu3ubuntu4',
}
package { 'apparmor':
  ensure => '2.10.95-0ubuntu2.9',
}
package { 'apport':
  ensure => '2.20.1-0ubuntu2.17',
}
```



LAB #3 - Gestion des Paquets

- Puppet resource a aussi un mode interactif :

```
vagrant@ubuntu-xenial:~$ sudo puppet resource -e package openssl
```

- Cette commande génère un manifest pour l'état actuel de la resource et l'ouvre dans un éditeur :

```
package { 'openssl':  
    ensure => '1.0.2g-1ubuntu4.12',  
}
```

- Si vous modifiez ce manifest, lors de l'enregistrement du fichier, Puppet appliquera les modifications

La gestion des services

- Utiliser la ressource de type service pour : démarrer, arrêter, activer ou désactiver un service
- La ressource service supporte plusieurs système de gestion de service (comme systemd, upstart, smf ...)
- Les paramètres de la ressource service :
 - Name : titre de la ressource ou le nom du service a gérer
 - Ensure : arrêter ou en execution
 - Enable : True / false



LAB #4 - Gestion des Services

- Créez le fichier suivant

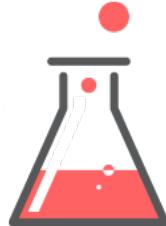
```
vagrant@ubuntu-xenial:~$ vi service.pp
vagrant@ubuntu-xenial:~$ cat service.pp
service { 'sshd':
  ensure => running,
  enable => true,
}
```

- Les attributs d'une ressource peuvent être consultés par la commande **puppet describe**

```
vagrant@ubuntu-xenial:~$ sudo puppet describe service
```

- Pour consulter la liste des types de ressources, utilisez la commande suivante :

```
vagrant@ubuntu-xenial:~$ sudo puppet describe --list
```



LAB #4 - Gestion des Services

- Les différents types de ressources peuvent être regroupés dans le même manifest.
 - Créez le fichier `package_file_service.pp` :
- Notez que dans ce manifest se trouve l'attribut `notify`. Cet attribut notifie le service `mysql` lors d'un changement de son fichier de configuration `mysql.cnf`.
 - L'action par défaut de Puppet dans ce cas est de redémarrer le service.
- L'ordre de la déclaration des ressources dans ce manifest est suivi par Puppet :
 - Puppet install le paquet `mysql-server`,
 - Puppet copie le fichier `/examples/files/mysql.cnf` vers `/etc/mysql/mysql.cnf`,
 - Puppet démarre le service `mysql`.

```
vagrant@ubuntu-xenial:~$ vi package_file_service.pp
vagrant@ubuntu-xenial:~$ cat package_file_service.pp
package { 'mysql-server':
  ensure => installed,
  notify => Service['mysql'],
}

file { '/etc/mysql/mysql.cnf':
  source => '/examples/files/mysql.cnf',
  notify => Service['mysql'],
}

service { 'mysql':
  ensure => running,
  enable => true,
}
```



LAB #4 - Gestion des Services

- Cet ordre est logique. Il est évident que le manifest suivant ne donnera pas un résultat satisfaisant :
- L'ordre de l'application des ressources dans un manifest peut cependant être fixé en utilisant l'attribut **require**.

- Voici le même manifest avec l'utilisation de cet attribut :

```
package { 'mysql-server':
  ensure => installed,
}

file { '/etc/mysql/mysql.cnf':
  source  => '/examples/files/mysql.cnf',
  notify  => Service['mysql'],
  require => Package['mysql-server'],
}

service { 'mysql':
  ensure  => running,
  enable   => true,
  require => [Package['mysql-server'], File['/etc/mysql/mysql.cnf']],
}
```

```
package { 'mysql-server':
  ensure => installed,
  notify => Service['mysql'],
}

service { 'mysql':
  ensure => running,
  enable => true,
}

file { '/etc/mysql/mysql.cnf':
  source => '/examples/files/mysql.cnf',
  notify => Service['mysql'],
}
```

Chapitre 5

Puppet et Git

61755866ee
Global Knowledge



- **LAB 10 : *La Ressource File***

- L'attribut Source
- L'Attribut owner
- L'Attribut group
- L'Attribut mode
- L'Attribut ensure
- L'Attribut recurse : Cet attribut permet de copier une arborescence complète.



- LAB #11 - La Ressource package
 - L'Attribut ensure
- LAB #12 - La Ressource service
 - L'Attribut hasstatus
 - L'Attribut pattern
 - Les Attributs hasrestart et restart

61755866ee
Global Knowledge



LAB #13 - La Ressource user

- Un utilisateur est un objet qui peut :
 - posséder des fichiers,
 - exécuter des commandes,
 - éventuellement peut lire ou modifier les fichiers d'autres utilisateurs
- *Créer un Utilisateur*
- *Supprimer un Utilisateur*



LAB #14 - La Ressource cron

- **L'Attribut user**

- Cet attribut spécifie qui exécute le cron job. Si ce n'est pas spécifié, le job est exécuté par root.

- **L'Attribut environment**

- Cet attribut permet de définir des variables système pour cron.

- **L'Attribut weekday**

- Cet attribut permet de spécifier les jours de la semaine.

- **L'Attribut monthday**

- Cet attribut permet de spécifier le jour du mois entre 1 et 31.

- **La Fonction fqdn_rand**

- Afin d'éviter à ce que les mêmes cron jobs ne s'exécutent sur tous les noeuds en même temps, Puppet utilise la fonction `fqdn_rand`
 - Cette fonction fournit un nombre aléatoire de 0 à 23 dans le cas de l'exemple ci-dessous :



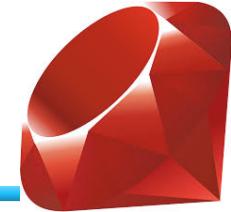
LAB #15 - La Ressource exec

- La ressource **exec** permet d'exécuter toute commande sur un noeud. Ce qui peut être exécuté en ligne de commande peut être exécuté en utilisant **exec**.
- **L'Attribut exec**
- **L'Attribut cwd**
 - Cet attribut fixe le répertoire de travail.
- **L'Attribut command**
 - Cet attribut indique la ou les commandes à exécuter. Il faut indiquer le chemin complet.
- **L'Attribut creates**
 - Cet attribut indique un fichier qui sera présent après l'exécution de la commande. Si ce fichier existe, Puppet n'exécutera pas la command une deuxième fois.
- **L'Attribut user**

- **L'Attribut *onlyif***
 - Certaines commandes ne doivent être exécutées que dans certaines conditions. Pour cette situation, Puppet possède l'attribut *onlyif* :
- **L'Attribut *unless***
 - L'attribut *unless* fait l'opposé de l'attribut *onlyif* à savoir, la commande sera toujours exécutée sauf si le code retour de la commande spécifiée par l'attribut *unless* est un 0.
- **L'Attribut *refreshonly***
- **L'Attribut *logoutput***
 - Cet attribut permet de modifier la journalisation de la commande.
- **L'Attribut *timeout***

Rappelle Ruby

- L'Histoire
 - Crée par Yukihiro Matsumoto (**Matz**)
 - Release publique en 2005
- C'est un langage
 - Plus objet que python
 - Plus puissant que perl
 - **Fun**



61755866ee
Global Knowledge

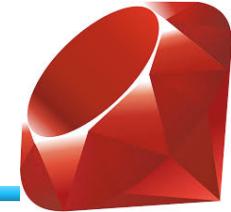
Le fameux hello world

Java

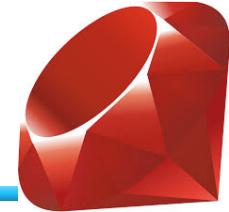
```
class Hello {  
    public static void main(String [] args)  
    {  
        system.out.println('Hello');  
    }  
}
```

Ruby

```
puts 'Hello'
```



Tout est objet



```
2.times { puts "Bonjour Supinfo." }
```

=> Bonjour Supinfo

=> Bonjour Supinfo

```
3.upto(5) { |i| puts i }
```

=> 3

=> 4

=> 5

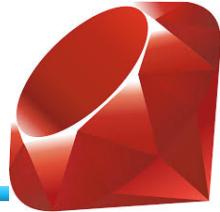
```
p l.zero?
```

=> false

```
39 + 3
```

=> 39.+ (3)

Les variables



```
class A
  MAX = 42
  @@instances = 0
  def initialize(name)
    @name = name
    @@instances += 1
  end

  def self.instances
    @@instances
  endend
```

foo
\$foo
@foo
@@foo
MAX_USERS

Local variable
Global variable
Instance variable in object
Class variable
Constant (by convention)

```
A.new("a")A.new("b")A.new("c")A.new("d")A.new("e")
p A.instances #=> 5
```

Itérations



```
a = 1..9  
for i in a  
  puts i  
end
```

=> 1...9

```
i = 0  
loop do i += 1  
  puts i  
  break if 10 == i  
end
```

=> 1...9

```
1.upto(2) do  
|i| v = rand(2)  
retry  
if v.zero?  
end
```

OU

```
a = 1..9  
a.each { |i| puts i }
```

=> 1...9

```
1.upto(10) do |i|  
  if i.odd? # pas d'impair en ruby  
    puts i  
  end  
end
```

=> 2, 4, 6, 8, 10

Conditions



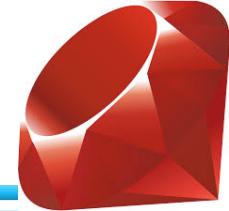
```
if index == 1  
else  
end
```

puts '0' if index.zero?

puts 'not 0' unless index.zero?

```
def what_is_it?(a)  
case a  
when 1..2  
puts "1 or 2"  
when 3  
puts "3"  
when /4.*#/ # Regexp  
puts "something starting with 4."  
when "foo"  
puts "foo"  
else puts "I don't know."  
end  
end
```

```
what_is_it?(1)# 1 or 2  
what_is_it?(2)# 1 or 2  
what_is_it?(3)# 3  
what_is_it?("4004")# something starting with 4.  
what_is_it?("foo")# foo  
what_is_it?(5)# Don't know.
```



Tableaux 1/2

1

```
lost = [8, 15, 16, 23]
lost << 42 # push
lost.unshift(4)
#[4, 8, 15, 16, 23,
42]
```

```
lost.at(0)
# 4
lost[-1]
# 42
```

```
lost << nil << nil
#[4, 8, 15, 16, 23,
42, nil, nil]
```

```
lost.flatten!.uniq!
#[4, 8, 15, 16, 23,
42]
```

```
lost << [4..8]
#[4, 8, 15, 16, 23,
42, [4..8]]
```

```
lost.compact!
#[4, 8, 15, 16, 23,
42]
```

```
lost.index(23)
# 4
```

```
lost.shuffle
#[16, 23, 42, 4, 15,
8]
[5, 3, 7, 39, 1,
15].sort
#[1, 3, 5, 7, 15, 39]
```

```
(a'..z).to_a
["a", "b", "c", "d"...]
```



Tableaux 2/2

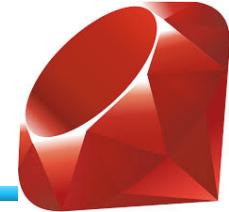
```
double_lost = lost.map { |v| v * 2 }
# => [8, 16, 30, 32, 46, 84]
```

```
# lost: [4, 8, 15, 16, 23, 42]
double_lost - lost
# => [30, 32, 46, 84]
```

```
# intersection
double_lost & lost
# [8, 16]
```

```
# jointure
(double_lost | lost).sort
# [4, 8, 15, 16, 23, 30, 32, 42, 46, 84]
```

String



```
str = "bar"  
puts 'foo #{str}'#  
=> foo #{str}
```

```
# Interpolation  
puts  
"foo #{str}"# =>  
foo bar
```

```
str = "a"str.succ#  
=> "b"
```

```
str =  
"foo\n"str.chomp#  
=> "foo"
```

```
str = "foo"str.chop#  
=> "fo"
```

```
"supinfo".capitalize  
# => "Supinfo"  
"supinfo".upcase  
# => "SUPINFO"
```

```
# attention, en 1.8  
str = "éhé"  
str.size# => 5 et non 3  
str[0]# 195 et non é (code  
ascii)
```

```
# patché dans rails  
str.mb_chars[0]  
# é  
str.mb_chars.size  
# 3
```

Hash

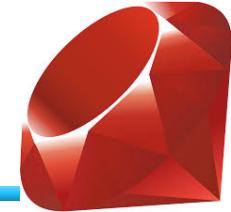
Association clé / valeur

```
h = { :a => 'a', :b => 42, :c => { :d => 'f' } }
```

```
h2 = { :a => 'foo' }
h.merge(h2)
=> { :a => "foo", :b => 42, :c => { :d => "f" } }
```



Class: les constructeurs

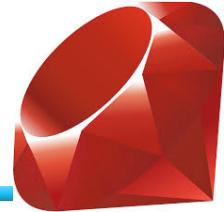


```
class A
  def A.new
  end
End
A.new
```

```
class B
  def self.new
  end
B.new
```

```
class C
  def initialize
  end
End*
C.new
```

Class: les accesseurs



```
class Product
  def initialize(name, description,
    price)
    @name = name
    @description = description
    @price = price
  end

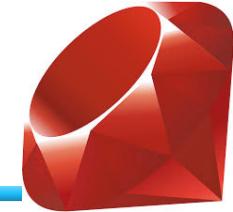
  def name
    @name
  end

  def name=(name)
    @name = name
  end
  ...
end
```

```
class Product
  attr_accessor :name
  attr_reader :description
  attr_writer :price

  def initialize(name, description,
    price)
    @name = name
    @description = description
    @price = price
  end
end
```

Class: portée & héritage

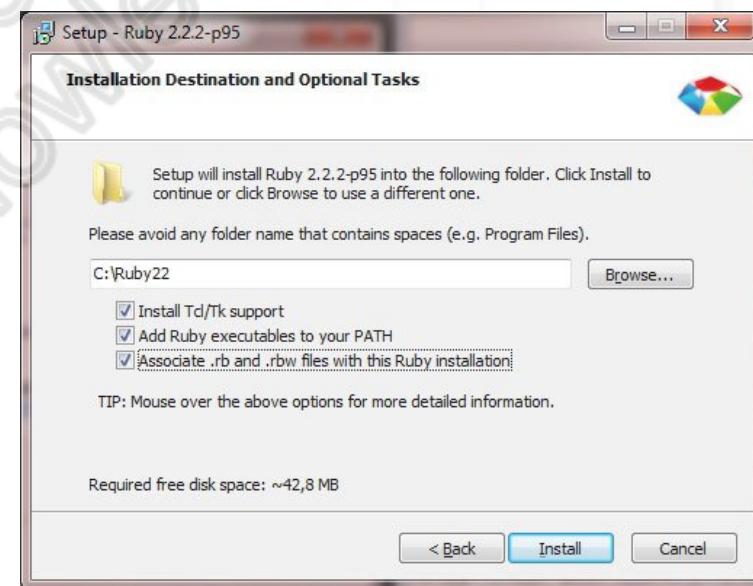
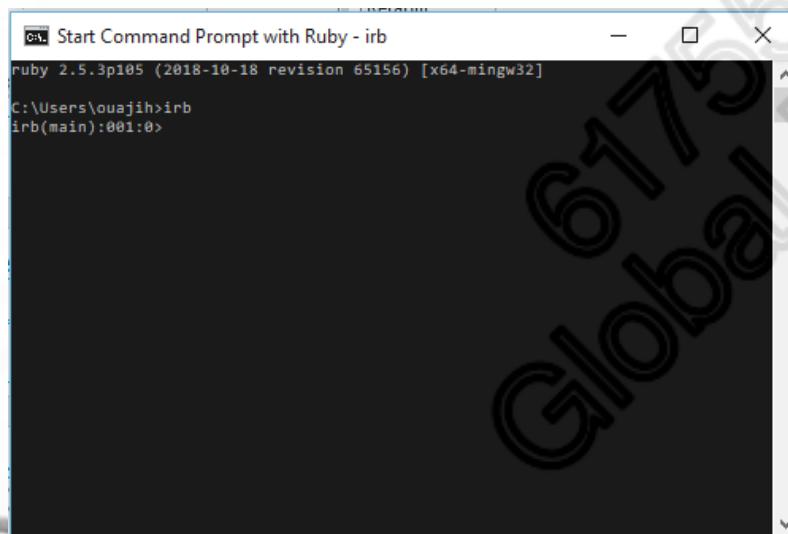


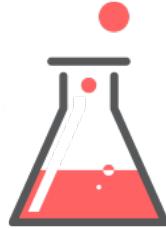
```
class Animal def initialize puts "Born  
to be alive." end protected def  
breathe? puts "inhale, exhale" true  
end private def speak; endend  
  
#Animal.new.speak  
#=> fail with private method `speak'
```

```
class Dog < Animal def alive? puts  
"I'm alive" if breathe? end def speak  
puts "woff." endend  
  
snoopy = Dog.new  
# Born to be alive.snoopy.speak  
# woff.snoopy.alive?  
# inhale, exhale# I'm alive
```

Workshop Ruby

- Rendez vous sur [Ruby Installer](https://rubyinstaller.org/) pour télécharger la dernière version de Ruby
 - <https://rubyinstaller.org/>
- Puis dans la console tapez irb





Variables, Expressions, Expressions Conditionnelles et Facts

- **LAB #16 – Variables**

- **Variables Simples**

- Le contenu de la variable peut être :
 - une chaîne,
 - un chiffre,
 - un booléen.

- **Tableaux**

- **Hashes**

- Un **Hash** est comme un Tableau, à l'exception du fait que chaque élément a un nom appelé une **clé**



Variables, Expressions, Expressions Conditionnelles et Facts

- **LAB #17 – Expressions**
 - Expressions Mathématiques
 - Expression Booléennes
 - Expressions Régulières
 - Expressions Conditionnelles

61755866ee
Global Knowledge



Variables, Expressions, Expressions Conditionnelles et Facts

- **LAB #18 – Facts**

Les manifests de Puppet ont souvent besoin de connaître quelque chose concernant le système, par exemple :

- Le nom d'hôte,
- L'adresse IP,
- La version du système d'exploitation.

Sous Puppet, le mécanisme qui permet d'obtenir ces informations s'appelle **Facter** et chaque information fournie par Facter s'appelle un **Fact**.

- Facts dans un Hash
- Facts dans une Expression
- Facts Externes
- Facts Exécutables



LAB #19 - Iteration

- *Iteration et Tableaux*
- *Iteration et Hashes*

61755866ee
Global Knowledge

Questions ?

61755866ee
Global Knowledge