Puppet version 4

Workshop France-Grilles 2016

Plan

Plan

Présentation générale de Puppet

- Gestionnaire de configurations
- Description de l'état souhaité (langage déclaratif)
- Code organisé en modules réutilisables
- Séparation du code et des données

Historique

- Fin de vie de la version 3
  - Fin du support après le 31 décembre 2016. (plus de correctifs sécurité pour la version 3)
  - `https://puppet.com/content/platform-support-lifecycle`
- Important de mettre à jour !

Plan

Outil facter

- fournit à Puppet une liste de variables spécifiques à l'agent à configurer
  - Exemples: *fqdn, kernelversion, operatingsystemrelease, . . .*
- Dans Puppet 4: table de hachage *"$facts"*
  - Elimine une source de confusion car une variable locale pouvait masquer une variable de facter (un "fact").
  - Permet de structurer les informations:

```
if $osfamily == 'RedHat' { ... }
=>
if $facts['os']['family'] == 'RedHat' { ... }
```

  - Permet de vérifier la présence d'une variable:

```
if (has_key($facts, 'myvar') && $facts['myvar'] == ...) { ... }
```

Outil Hiera

- Fournit les paramètres des classes à Puppet.
- Séparation du code et des données
- Base de données hiérarchique (sur le puppet master)
- Hiérarchie configurable
- Un ensemble structuré de fichiers contient les données
- Possibilité d'encrypter les données sensibles
- Installé par défaut avec Puppet 4

Nouveaux emplacements des fichiers de configuration

Executables	/opt/puppetlabs/bin/
Configuration	/etc/puppetlabs/puppet/
Modules	/etc/puppetlabs/code/modules/
Environnements	/etc/puppetlabs/code/environments/
Manifestes	/etc/puppetlabs/code/environments/production/manifests/

Environnements

- Permet de partitionner les agents en sous-ensembles
  - Exemples: production, test, ...
- Chaque agent décide de son environnement via son fichier de configuration: L'agent à configurer va demander au puppet master le catalogue correspondant à cet environnement.
- Avec Puppet 4: organisé par répertoires sur le puppet master

```
/etc/puppetlabs/code/environments/test/
/etc/puppetlabs/code/environments/test/manifests
/etc/puppetlabs/code/environments/test/modules

/etc/puppetlabs/code/environments/production/
/etc/puppetlabs/code/environments/production/manifests
/etc/puppetlabs/code/environments/production/modules
```

Typage des données

Puppet 3: avec la stdlib

```
class myclass(
 $var,
 $array,
 $paths
) {
 validate_string($var)
 validate_array($array)
 validate_absolute_path($paths)
 ...
}
```

mais deprecated : "Warning: This method is deprecated"

Typage des données

- avec Puppet 4: Annotations de typage
- Un nom de type commence par une majuscule: String, Integer, Boolean, Variant, Hash, etc.

```
class myclass(
 String $var,
 Array[Integer] $ports,
 Array[Pattern[/^\//]] $paths,
) {
 ...
}
```

Typage des données

Définir ses propres types:

```
type AbsolutePath = Pattern[/^\//]

class myclass(
 String $var,
 Array[Integer] $ports,
 Array[AbsolutePath] $paths,
) {
 ...
}
```

Nouvelles opérations sur les données.

- Tester une sous-chaîne:

```
$str1 in $str2
```

- Extraction de sous-chaîne:

```
"xyzabcdef"[3,3] # : "abc"
```

- Fusion de données:

```
$v1 = [0,2]; $v2 = [1,3]; $v1 + $v2 # : [0,2,1,3]

$x = {"a" => "aa"}; $y = {"b" => "bb"};
$x + $y # : {"a" => "aa", "b" => "bb"}

[1,2,3,4] << 5 # : [1,2,3,4,5]

$a = [2,3]; $b = [1, *$a, 4] # : [1,2,3,4]
```

Paramètres par défaut dans les ressources

```
file {
  default:
    mode   => '0600',
    owner  => 'root',
    group  => 'root',
    ensure => file,
  ;
  '/etc/ssh/ssh_host_key': ;
  '/etc/ssh/ssh_host_dsa_key.pub':
    mode => '0644',
}
```

Itérations

A partir de données (Array, Hash, etc)

- **each** avec un Array:

```
["a", "b", ...].each |$file| { ... }
```

- **each** avec un Hash:

```
$myhashtable.each |$key, $value| { ... }
```

- **map**: applique le code sur chaque valeur et retourne le résultat.

```
$table = ["a", "b", ...]
$data = $table.map |$var| { ... }
```

- **filter**: ne conserve que les élements selectionnés

```
$data = [1,2,3,4].filter |$i| { $i > 2 } # : [3,4]
```

- **reduce**: applique itérativement un calcul

```
$data = [1, 2, 3]
$sum = $data.reduce |$memo, $value| { $memo + $value }
# : (1 + 2) + 3 = 6
```

Plan

Nouveau parseur, nouvelle syntaxe

- Ne pas inclure '-' dans les noms de classe, pas de majuscule ni de '_' en début de nom, etc.
  - `https://docs.puppet.com/puppet/4.8/reference/lang_reserved.html`
- Suppression des import: utiliser des include à la place, suppression de l'héritage.
- **Avec Puppet 4 les assignements sont maintenant interdits**: le contenu des tableaux ou des tables de hachage n'est pas modifiable.

```
$config_class['nagios::nrpe'] = $config_hash

  Illegal attempt to assign via [index/key].
```

Nouvelle portée des variables dans les modules

- Puppet 4: la résolution dynamique des variables n'existe plus.
- Exemple:
  - La variable *$devicetype* est un paramètre de la classe cups:

```
class cups(
  $devicetype = 'Server',
  $server
) {
    include cups::config, cups::service, cups::install
}
```

  - mais cette variable n'est pas visible dans cups::service !
    - Ok avec Puppet 3
  - Il suffit d'y accéder globalement dans cups::service via cups::devicetype.

Portée des variables dans les templates

De même avec l'accès aux variables dans les templates: il faut le chemin complet et utiliser *scope['...']*.

```
<% @volumes.each do |volume| -%>
<%= @description[volume]["mountname"] %>
<%end -%>

  undefined method `[]' for nil:NilClass

=>

<% @volumes.each do |volume| -%>
<%= scope['autofs::description'][volume]["mountname"] %>
<%end -%>
```

Expressions régulières

Ne fonctionnent maintenant que sur du texte:

```
  $level = 2

  case $level {
    /[1-3]/: { notify { 'low': } }
    /[4-7]/: { notify { 'medium': } }
    default: { notify { 'high': } }
  }

Puppet 3.x => 'low'
Puppet 4.x => 'high'
```

Plan

Conclusion

- Beaucoup de nouveautés pratiques
- Attention aux modules externes non compatible avec la version 4 !
  - *puppet-xrootd*, *puppet-cvmfs*, *puppet-bdii*, etc. utilisent l'héritage de classe (non compatible avec Puppet 4)
- Il est nécessaire d'adapter tout son code (templates, modules, manifests) entre la version 3 et la version 4.

Merci !

https://docs.puppet.com/