Version : **2022.01**

Dernière mise-à-jour : 2022/05/26 15:12

LDF803 - Variables, Expressions, Facts et Itérations

Contenu du Module

LAB #1 - Variables

Variables Simples

Une variable sous Puppet est une façon de donner un nom à une valeur. Par exemple :

```
$php_package = 'php7.0-cli'

package { $php_package:
  ensure => installed,
}
```

Important - Le nom d'une variable doit commencer par le caractère **$** puis par une lettre minuscule ou un underscore.

Le contenu de la variable peut être :

- une chaîne,
- un chiffre,
- un booléen.

```
$my_name = 'Zaphod Beeblebrox'
$answer = 42
$scheduled_for_demolition = true
```

Important - La valeur d'un variable booléenne doit être **true** ou **false**.

Les variables peuvent ensuite être appelées ainsi :

```
vagrant@ubuntu-xenial:~$ sudo vi string_interpolation.pp
vagrant@ubuntu-xenial:~$ cat string_interpolation.pp
$my_name = 'John'
notice("Hello, ${my_name}! It's great to meet you!")

vagrant@ubuntu-xenial:~$ sudo puppet apply string_interpolation.pp
Notice: Scope(Class[main]): Hello, John! It's great to meet you!
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Important - Notez l'utilisation de l'attribut **notice** qui imprime la chaîne et le contenu de la variable à l'écran.

Tableaux

Un tableau est une groupe de valeurs. Par exemple :

```
$heights = [193, 120, 181, 164, 172]

$first_height = $heights[0]
```

Il est possible de référencer une des valeurs en utilisant son index. Par exemple **$heights[0]=193** et **$heights[4]=172**.

Dans l'exemple de l'utilisation d'un tableaux ci-dessous, le tableau contient une liste de dépendances qui doivent être installées sur le système :

```
vagrant@ubuntu-xenial:~$ sudo vi resource_array.pp
vagrant@ubuntu-xenial:~$ cat resource_array.pp
$dependencies = [
```

```
  'php7.0-cgi',
  'php7.0-cli',
  'php7.0-common',
  'php7.0-gd',
  'php7.0-json',
  'php7.0-mcrypt',
  'php7.0-mysql',
  'php7.0-soap',
]

package { $dependencies:
  ensure => installed,
}
```

Appliquez ce manifest :

```
vagrant@ubuntu-xenial:~$ sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:3 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Hit:5 http://apt.puppetlabs.com xenial InRelease
Fetched 325 kB in 0s (472 kB/s)
Reading package lists... Done

vagrant@ubuntu-xenial:~$ sudo puppet apply resource_array.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.48 seconds
Notice: /Stage[main]/Main/Package[php7.0-cgi]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-gd]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-mcrypt]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-mysql]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-soap]/ensure: created
Notice: Applied catalog in 16.33 seconds
```

Comme vous pouvez constater, Puppet a créé une ressource pour chaque paquet qui doit être installé :

```
Notice: /Stage[main]/Main/Package[php7.0-cgi]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-gd]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-mcrypt]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-mysql]/ensure: created
Notice: /Stage[main]/Main/Package[php7.0-soap]/ensure: created
```

Hashes

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

```
vagrant@ubuntu-xenial:~$ sudo vi variable_hash.pp
vagrant@ubuntu-xenial:~$ cat variable_hash.pp
$heights = {
  'john'    => 193,
  'rabiah'  => 120,
  'abigail' => 181,
  'melina'  => 164,
  'sumiko'  => 172,
}

notice("John's height is ${heights['john']}cm.")
```

Appliquez ce manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply variable_hash.pp
Notice: Scope(Class[main]): John's height is 193cm.
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Un Hash peut être utilisé pour définir les attributs d'une ressource :

```
vagrant@ubuntu-xenial:~$ sudo vi hash_attributes.pp
vagrant@ubuntu-xenial:~$ cat hash_attributes.pp
$attributes = {
  'owner' => 'ubuntu',
  'group' => 'ubuntu',
  'mode'  => '0644',
}

file { '/tmp/test':
  ensure => present,
  *      => $attributes,
}
```

Important - Le caractère * informe Puppet d'utiliser ce Hash comme une liste d'attributs.

Appliquez ce manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply hash_attributes.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: /Stage[main]/Main/File[/tmp/test]/ensure: created
Notice: Applied catalog in 0.01 seconds

vagrant@ubuntu-xenial:~$ ls -l /tmp/test
-rw-r--r-- 1 ubuntu ubuntu 0 Feb 11 13:42 /tmp/test
```

LAB #2 - Expressions

Expressions Mathématiques

Puppet peut gérer des expressions mathématiques. Créez donc le fichier **expression_numeric.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi expression_numeric.pp
vagrant@ubuntu-xenial:~$ cat expression_numeric.pp
$value = (17 * 8) + (12 / 4) - 1
notice($value)
```

L'application de ce manifest affichera le résultat de l'expression :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply expression_numeric.pp
Notice: Scope(Class[main]): 138
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Expression Booléennes

Puppet sait aussi gérer des expression booléennes, c'est-à-dire des expressions qui produisent comme résultat soit **vrai** soit **faux**. Créez donc le fichier **expression_boolean.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi expression_boolean.pp
vagrant@ubuntu-xenial:~$ cat expression_boolean.pp
notice(9 < 10)
notice(11 > 10)
notice(10 >= 10)
notice(10 <= 10)
notice('foo' == 'foo')
notice('foo' in 'foobar')
notice('foo' in ['foo', 'bar'])
notice('foo' in { 'foo' => 'bar' })
notice('foo' =~ /oo/)
```

```
notice('foo' =~ String)
notice(1 != 2)
```

Les opérateurs utilisés dans le fichier **expression_boolean.pp** sont les suivants :

Opérateur	**Description**
==	Égal
!=	Pas égal
>, >=, < et <=	Supérieur, supérieur ou égal, inférieur, inférieur ou égal
A in B	A est une sous-chaîne de B, A est un élément du tableau B ou A est une clef du Hash B
A =~ B	A correspond à l'expression régulière B ou A est une valeur qui correspond au type de données de B (p.e. **'trainee' =~ String** est vrai)
Valeur =~ ER	Si valeur égal l'ER alors vrai
/a+/	Correspond à a aa aaa etc

Les expressions dans le fichier **expression_boolean.pp** donnent toutes un résultat de **vrai**, comme vous pouvez voir en appliquant le manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply expression_boolean.pp
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Expressions Régulières

Les expression régulières dans Puppet doivent être entourées du caractère **/**. Créez don le fichier **regex.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi regex.pp
vagrant@ubuntu-xenial:~$ cat regex.pp
$candidate = 'foo'
notice($candidate =~ /foo/) # literal
notice($candidate =~ /f/)   # substring
notice($candidate =~ /f.*/) # f followed by zero or more characters
notice($candidate =~ /f.o/) # f, any character, o
notice($candidate =~ /fo+/) # f followed by one or more 'o's
notice($candidate =~ /[fgh]oo/) # f, g, or h followed by 'oo'
```

Comme la valeur contenue dans **$candidate** correspond à chaque expression régulière dans le manifest, le résultat est toujours **vrai** :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply regex.pp
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Scope(Class[main]): true
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Important - La syntaxe des expressions régulières Puppet est la même que celle de **Ruby**. Vous pouvez trouver plus d'informations concernant cette syntaxe à l'adresse suivante : **http://ruby-doc.org/core/Regexp.html**.

Expressions Conditionnelles

Puppet peut utiliser des expressions. Créez le fichier **if.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi if.pp
vagrant@ubuntu-xenial:~$ cat if.pp
$install_perl = true
if $install_perl {
  package { 'perl':
    ensure => installed,
  }
} else {
  package { 'perl':
    ensure => absent,
  }
}
```

Appliquez ce menifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply if.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.55 seconds
Notice: Applied catalog in 0.04 seconds
```

Contrôlez maintenant si la commande **dpkg –get-selections** retourne une valeur d'**install** pour le paquet **perl** :

```
vagrant@ubuntu-xenial:~$ dpkg --get-selections | grep perl
libapparmor-perl                install
liberror-perl                   install
liblocale-gettext-perl              install
libperl5.22:amd64               install
libtext-charwidth-perl              install
libtext-iconv-perl              install
libtext-wrapi18n-perl               install
```

```
perl                        install
perl-base                   install
perl-modules-5.22               install
```

Modifiez maintenant la valeur de **$install_perl** dans le fichier **if.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi if.pp
vagrant@ubuntu-xenial:~$ cat if.pp
$install_perl = false
if $install_perl {
  package { 'perl':
    ensure => installed,
  }
} else {
  package { 'perl':
    ensure => absent,
  }
}
```

Appliquez maintenant le manifest modifié :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply if.pp
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.46 seconds
Notice: /Stage[main]/Main/Package[perl]/ensure: removed
Notice: Applied catalog in 2.03 seconds
```

Contrôlez maintenant si la commande **dpkg –get-selections** retourne une valeur d'**deinstall** pour le paquet **perl** :

```
vagrant@ubuntu-xenial:~$ dpkg --get-selections | grep perl
libapparmor-perl                install
liblocale-gettext-perl              install
libperl5.22:amd64               install
libtext-charwidth-perl              install
libtext-iconv-perl              install
```

```
libtext-wrapi18n-perl               install
perl                          deinstall
perl-base                     install
perl-modules-5.22                 install
```

HERE

Une expression conditionnelle utilisant un **if** ne permet que deux choix. S'il faut plus de choix, il convient d'utiliser **case**. Créez donc le fichier **case.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi case.pp
vagrant@ubuntu-xenial:~$ cat case.pp
$webserver = 'nginx'
case $webserver {
  'nginx': {
    notice("Looks like you're using Nginx! Good choice!")
  }
  'apache': {
    notice("Ah, you're an Apache fan, eh?")
  }
  'IIS': {
    notice('Well, somebody has to.')
  }
  default: {
    notice("I'm not sure which webserver you're using!")
  }
}
```

En appliquant ce manifest, Puppet retourne la chaîne **Looks like you're using Nginx! Good choice!** :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply case.pp
Notice: Scope(Class[main]): Looks like you're using Nginx! Good choice!
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
```

```
Notice: Applied catalog in 0.01 seconds
```

LAB #3 - 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

Créez donc le fichier **facts_hash.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi facts_hash.pp
vagrant@ubuntu-xenial:~$ cat facts_hash.pp
notice($facts['kernel'])
```

Important - Dans ce manifest est utilisé la variable **$facts**. Le fact recherché est fourni en tant que clef - dans notre exemple **kernel**.

En appliquant ce manifest, vous obtiendrez donc le type de noyau du système d'exploitation :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply facts_hash.pp
Notice: Scope(Class[main]): Linux
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
```

```
Notice: Applied catalog in 0.01 seconds
```

L'utilisation de la commande **facter** permet de se renseigner sur les Facts disponibles, par exemple ceux concernant le système d'exploitation :

```
vagrant@ubuntu-xenial:~$ facter os
{
  architecture => "amd64",
  distro => {
    codename => "xenial",
    description => "Ubuntu 16.04.6 LTS",
    id => "Ubuntu",
    release => {
      full => "16.04",
      major => "16.04"
    }
  },
  family => "Debian",
  hardware => "x86_64",
  name => "Ubuntu",
  release => {
    full => "16.04",
    major => "16.04"
  },
  selinux => {
    enabled => false
  }
}
```

Important - Comme vous pouvez constater, la sortie est sous forme de Hashs multiples.

Pour accéder à une valeur dans un Hash, il convient de spécifier la clef contenant la valeur entre les caractères **[ ]**. Pour spécifier une clef dans une clef, il convient de nouveau à la spécifier entre des caractères **[ ]**. Créez le fichier **facts_architecture.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi facts_architecture.pp
vagrant@ubuntu-xenial:~$ cat facts_architecture.pp
notice($facts['os']['architecture'])
```

Appliquez le manifest et vous obtenez la valeur d'**architecture** :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply facts_architecture.pp
Notice: Scope(Class[main]): amd64
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Cette même technique est applicable quelque soit le profondeur du Hash. Pour obtenir la valeur du **codename**, créez le fichier **facts_distro_codename.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi facts_distro_codename.pp
vagrant@ubuntu-xenial:~$ cat facts_distro_codename.pp
notice($facts['os']['distro']['codename'])
```

Appliquez le manifest et vous obtenez la valeur **xenial** :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply facts_distro_codename.pp
Notice: Scope(Class[main]): xenial
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Pour obtenir la valeur du numéro de version de la distribution, créez le fichier **facts_major** :

```
vagrant@ubuntu-xenial:~$ sudo vi facts_major.pp
vagrant@ubuntu-xenial:~$ cat facts_major.pp
notice($facts['os']['release']['major'])
```

Appliquez le manifest et vous obtenez la valeur **16.04** :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply facts_major.pp
Notice: Scope(Class[main]): 16.04
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Pour obtenir le nom d'hôte, le FQDN et l'adresse IP de la machine, créez le fichier **fact_networking.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi fact_networking.pp
vagrant@ubuntu-xenial:~$ cat fact_networking.pp
notice("My hostname is ${facts['hostname']}")
notice("My FQDN is ${facts['fqdn']}")
notice("My IP is ${facts['networking']['ip']}")
```

Appliquez le manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply fact_networking.pp
Notice: Scope(Class[main]): My hostname is ubuntu-xenial
Notice: Scope(Class[main]): My FQDN is ubuntu-xenial
Notice: Scope(Class[main]): My IP is 10.0.2.15
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Facts dans une Expression

Il est aussi possible de référencer un Fact dans une expression. Créez don le fichier **fact_ip.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi fact_ip.pp
vagrant@ubuntu-xenial:~$ cat fact_ip.pp
if $facts['os']['selinux']['enabled'] {
  notice('SELinux is enabled')
```

```
} else {
  notice('SELinux is disabled')
}
```

Appliquez le manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply fact_ip.pp
Notice: Scope(Class[main]): SELinux is disabled
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Facts permettent la configuration dynamique de variables pour les applications. Dans le cas suivant, on va configurer la variable MariaDB **innodb_buffer_pool_size**. Créez donc le fichier **fact_memory.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi fact_memory.pp
vagrant@ubuntu-xenial:~$ cat fact_memory.pp
$buffer_pool = $facts['memory']['system']['total_bytes'] * 3/4
notice("innodb_buffer_pool_size=${buffer_pool}")
```

En appliquant ce manifest la valeur de la directive **innodb_buffer_pool_size** est fixée à **75%** de la mémoire RAM totale :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply fact_memory.pp
Notice: Scope(Class[main]): innodb_buffer_pool_size=780091392
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Facts Externes

Facts Externes sont des Facts créés par l'Administrateur. Puppet cherche des Facts Externes dans le répertoire **/opt/puppetlabs/facter/facts.d/**. Créez le fichier **fact_external.txt** et copiez-le dans le répertoire **/opt/puppetlabs/facter/facts.d/** :

```
vagrant@ubuntu-xenial:~$ sudo vi fact_external.txt
```

```
vagrant@ubuntu-xenial:~$ cat fact_external.txt
cloud=aws
vagrant@ubuntu-xenial:~$ sudo cp fact_external.txt /opt/puppetlabs/facter/facts.d/
```

Utilisez ensuite la commande **facter** pour vérifier que le Fact fonctionne :

```
vagrant@ubuntu-xenial:~$ sudo facter cloud
aws
```

Dans un manifest, un Fact Externe est référencé exactement de la même façon qu'un Fact Interne. Créez le fichier **fact_cloud.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi fact_cloud.pp
vagrant@ubuntu-xenial:~$ cat fact_cloud.pp
case $facts['cloud'] {
  'aws': {
    notice('This is an AWS cloud server ')
  }
  'gcp': {
    notice('This is a Google cloud server')
  }
  default: {
    notice("I'm not sure which cloud I'm in!")
  }
}
```

Appliquez le manifest pour obtenir le résultat :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply fact_cloud.pp
Notice: Scope(Class[main]): This is an AWS cloud server
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.01 seconds
Notice: Applied catalog in 0.01 seconds
```

Facts Exécutables

Facts Externes ne sont pas uniquement des fichiers textes statiques. Ils peuvent aussi être la sortie d'un script ou d'une commande. Créez le script **date.sh** et copiez-le dans le répertoire **/opt/puppetlabs/facter/facts.d/** :

```
vagrant@ubuntu-xenial:~$ sudo vi date.sh
vagrant@ubuntu-xenial:~$ cat date.sh
#!/bin/bash
echo "date=`date +%F`"
vagrant@ubuntu-xenial:~$ sudo cp date.sh /opt/puppetlabs/facter/facts.d/
```

Rendez le script exécutable et vérifiez avec la commande **facter** que le Fact fonctionne :

```
vagrant@ubuntu-xenial:~$ sudo chmod a+x /opt/puppetlabs/facter/facts.d/date.sh
vagrant@ubuntu-xenial:~$ sudo facter date
2020-02-11
```

LAB #4 - Itérations

Itération permet d'économiser du code.

Itération et Tableaux

Créez le fichier **iteration_simple.pp**

```
vagrant@ubuntu-xenial:~$ sudo vi iteration_simple.pp
vagrant@ubuntu-xenial:~$ cat iteration_simple.pp
file { '/usr/local/bin/task1':
  content => "echo I am task1\n",
  mode    => '0755',
```

```
}

file { '/usr/local/bin/task2':
  content => "echo I am task2\n",
  mode    => '0755',
}

file { '/usr/local/bin/task3':
  content => "echo I am task3\n",
  mode    => '0755',
}
```

Important - Dans ce manifest il y a trois ressources presque identiques, différenciées uniquement par le numéro de tâche (**task1**, **task2**, **task3**).

Créez maintenant le fichier **iteration_each.pp** :

```
vagrant@ubuntu-xenial:~$ sudo vi iteration_each.pp
vagrant@ubuntu-xenial:~$ cat iteration_each.pp
$tasks = ['task1', 'task2', 'task3']
$tasks.each | $task | {
  file { "/usr/local/bin/${task}":
    content => "echo I am ${task}\n",
    mode    => '0755',
  }
}
```

Important - Dans ce manifest la fonction **each** crée une boucle qui prend pour chaque exécution une des valeurs du tableau **$tasks** (**task1**, **task2**, **task3**).

La forme du boule **each** est :

```
ARRAY.each | ELEMENT | {
BLOCK
}
```

Important - Ici ARRAY est un tableau et ELEMENT est le nom d'une variable qui contiendra chaque valeur du tableau à tour de rôle. BLOCK est du code Puppet qui appelle l'ELEMENT.

Iteration et Hashes

Créez le fichier suivant :

```
vagrant@ubuntu-xenial:~$ sudo vi iteration_hash.pp
vagrant@ubuntu-xenial:~$ cat iteration_hash.pp
$nics = $facts['networking']['interfaces']
$nics.each | String $interface, Hash $attributes | {
  notice("Interface ${interface} has IP ${attributes['ip']}")
}
```

Appliquez le manifest :

```
vagrant@ubuntu-xenial:~$ sudo puppet apply iteration_hash.pp
Notice: Scope(Class[main]): Interface enp0s3 has IP 10.0.2.15
Notice: Scope(Class[main]): Interface lo has IP 127.0.0.1
Notice: Compiled catalog for ubuntu-xenial in environment production in 0.02 seconds
Notice: Applied catalog in 0.01 seconds
```

La liste des interfaces réseau retournée est un Hash nommé **$nics**.
