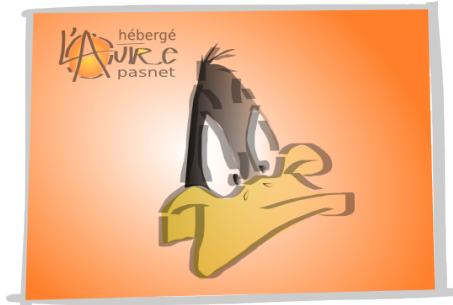


<http://www.coincoin.fr.eu.org/?Iterating-in-Puppet>



# Iterating in Puppet

- 6- Webographie -

Date de mise en ligne : mercredi 16 décembre 2015

---

**Copyright © L'Imp'Rock Scénette (by @\_daffyduke\_) - Tous droits réservés**

---

Iteration in Puppet has been a long standing pain point, Puppet 4 address this by adding blocks, loops etc. Here I capture the various approaches to working with some complex data in Puppet before and after Puppet 4

To demonstrate this I'll take some data from a [previous blog post](#) and see how to deal with it, here's the data that will be in `$domains` in the examples blow : `"x.net" : "nexthop" : "70.x.x.x", "spamdestination" : "rip@devco.net", "spamthreshold" : 1500, "enable_antispam" : 1 , "x.co.uk" : "nexthop" : "70.x.x.x", "spamdestination" : "rip@devco.net", "spamthreshold" : 1500, "enable_antispam" : 1 ,`

First we're going to need some defined type that can create an individual domain, we'll call that `mail::domain` but I won't show the code here, as that's not really important.

## Puppet 3 + stdlib

The first approach I'll show your basic Puppet 3 approach. The basic idea here is to get a list of domains and use the array iteration Puppet has always had on name.

The trick here is to get the domain names using the `keys()` function and then pass all the data into every instance of the define - the instance fetch it's data from the data passed into the define. `$domain_names = keys($domains)`  
`mail::domains$domain_names : domains => $domains`  
`define mail::domains($domains) $domain =`  
`$domains[$name] mail::domain$name : nexthop => $domain["nexthop"] . .`

## Puppet 3 + create\_resources

A hacky riff on `eval()` was added to Puppet during 3 to make it a bit easier to deal with data from Hiera or similar, it takes some data in a standard format and create instances of a defined type : `create_resources("mail::domain", $domains, "spamthreshold" => 1500, "enable_antispam" => 1)`

This replaces all the code above plus adds some default handling in the case that the data is not uniform. Some people love it, some hate it, I think it's a bit too magical so prefer to avoid it.

## Puppet 4 - each loop

This is the approach you'd probably want to use in Puppet 4 it uses a simple each loop over the data :  
`$domains.each |$name, $domain| mail::domain$name : nexthop => $domain["nexthop"] . .`

It's quite readable and obvious what's happening here, it's more typing than the `create_resources` example but I think this is the preferred way due to clarity etc

Below this we get into the more academic solutions to the problem, mainly showing off some Puppet 4 features.

## Puppet 4 - wildcard shortcut

If listing every key is tedious like above and if you know your hashes map 1:1 to the defined type parameters you can short circuit things a bit, this is quite close to the `create_resources` convenience : `each($domains) |$name, $domain| mail::domain$name : * => $domain`

The splat operator takes all the data in the hash and maps it right onto properties of the define type, quite handy

## Puppet 4 - wildcard and defaults

Your data might not all be complete so you'd want to get some defaults merged in, this is something `create_resources` also supports so this is how you'd do it without `create_resources` : `$defaults = "spamthreshold" => 1500, "enable_antispam" => 1 $domains.each |$name, $domain| mail::domain$name : * => $defaults + $domain # + now merge hashes`

## Puppet 4 - wildcard and resource defaults

An alternative to the above that's a bit more verbose but might be more readable can be seen below : `$defaults = "spamthreshold" => 1500, "enable_antispam" => 1 $domains.each |$name, $domain| mail::domain default : * => $defaults ; $name : * => $domain`

## Conclusion

That's about it, there are many more iteration tricks in Puppet 4 but this shows you how to achieve what you did with `create_resources` in the past and a couple of possible approaches to solving that problem.

Not sure which I'd recommend, but I suspect the choice comes down to personal style and situation.