Puppet @ PIC

Bruno Rodríguez
Legacy: from puppet 2 to puppet 5

- We have used Puppet at PIC for a long time.
  - this is the second presentation PIC makes about puppet...
  - ... The first one was in 2010, so quite a lot of things have changed

- From puppet 2 to 3 we made only minor code changes
  - Mainly started to use git (gitolite) instead of SVN
  - Still using local scripts/hooks (r10k didn't fit in our old code)

- Puppet 4 involved a total redesign that took a lot of time
  - thinking a hiera schema was difficult because of the already working code
  - puppet users also manage other (critical) services
    - migrating puppet code was a secondary task
  - puppet is considered "a configuration tool"
    - reluctance to use Puppet forge modules: sensation of losing control
The actual picture

- **Two servers, puppet 4 and puppet 5**
  - The code is mirrored between both, it's 100% compatible
  - Physical servers, 32 GB RAM, 2x1Gbps (active/passive)
  - ~720 clients: WNs, UIs, dcache, voms, squids... Mostly everything but oVirt hypervisors

- **Environments managed with r10k and g10k**
  - Puppetfile has local modules, puppet forge modules and some forks of them (in github)
  - g10k is way faster because of go instead of ruby and module caching

- **Local repositories managed with gitolite**
  - Lightweight, simple, relatively easy to configure
  - Allows user/permission management, local hooks, mirroring
  - can be interfaced via httpd server (with openldap integration)
Puppet code deployment

Inside the puppet master

Apache + LDAP

HTTP interface to

Gitolite

Checks syntax

pre-receive hook

If syntax OK

push to mirrors

Invokes

post-receive hook

If syntax OK

Gets code from

r10k / g10k

Deploys

/./etc/puppetlabs/code/

Mirrors

puppet forge

Github
My (our) puppet philosophy

This points helped keeping (part of) my sanity while dealing with puppet code

- Create custom facts that can be applied to your hierarchy
  - A few Ruby lines can solve a problem requiring lots of puppet code
- If possible, use external code (approved by puppet)
  - Better and more scalable than the "simple code I want to do just for this"
- Declare as much as possible parameters via hiera
  - YAML is easier to read and puppet code errors lead to bigger problems
  - We have "wrapper" classes to call `create_resources` defines from external modules
- KISS (keep it simple!)
  - Sometimes a new cool feature only adds unneeded complexity
A point about security

In our approach, sensitive data is only as safe as the access to the git repo where we keep our yaml files.

We use `eyaml` with the GPG backend to encrypt our values. Using GPG was a legacy decision that works OK for us:

- GPG is well known and defaultly available for most distros
- No need to install either ruby and the ruby gem on your workstation
- That means "your workmates stations" too
  - I won't blame anyone that does not want to install ruby + gems just to encrypt values
Wrappers

A small example of a wrapping class (**vcsrepo** gets a remote repository)

class wrapper::vcsrepo (  
  Hash $vcsrepos = {},  
) {  
  create_resources('vcsrepo', $vcsrepos)  
}

This would be called in hiera like

classes:
  # ...
  - wrapper::vcsrepo

wrapper::vcsrepo::vcsrepos:
  '/path/where/I/want/the/repo':
    source: 'ssh://username@example.com/repo.git'
    user: 'mrajoy' # uses mrajoy's $HOME/.ssh setup
This Ruby code will set the fact `pic_common['service']` for a Kafka server.

```ruby
require 'facter'
require 'socket'

hostname = Socket::gethostname

Facter.add(:pic_common) do
  setcode do
    case
    when match=hostname.match(/^kafka(\d+)\.(pic\.)es$/)
      pic_common['service'] = 'kafka'
      pic_common['num_server'] = match.captures[0]
      pic_common['collectd']['cluster'] = 'Kafka'
      ...
    end
    pic_common
  end
end
```

- **Service name**
  - 01 for kafka01.pic.es, 02 for kafka02.pic.es, etc.
  - Useful to define server id in zookeeper

- **Other service information** (monitoring, collectd, etc.)
We use the previous fact in the `/etc/puppetlabs/puppet/hiera.yaml`

```yaml
version: 5
defaults:
  datadir: "\!/etc/puppetlabs/code/environments/%{environment}/hieradata"
  lookup_key: eyaml_lookup_key
  options:
    gpg_gnupghome: '/\!/etc/puppetlabs/puppet/gpg_keys'
    encrypt_method: 'gpg'

:hierarchy:
- name: "Per node data"
  path: "nodes/%{::trusted.certname}.yaml"
- name: "Per service"
  path: "services/%{::pic_common.service}/defaults.yaml"
# ...
- name: "Last resort values"
  path: "common.yaml"
```
Possible EGI improvements

As we said, we have been using Puppet for a long time now but I'm afraid that YAIM is still our "elephant in the room"

- What are YAIM future plans?
- We are not solving the problem in a very different way than 10 years ago
- I only found a puppet-emi module (thank you, Alessandro) made around 7 years ago