Writing modules and reading manifests

DEVELOPING WITH PUPPET

What a PT has to do

1. Write puppet module per component
2. Test puppet module
   - Together with other modules
3. Publish puppet modules
   - Public git repo
   - puppetforge

One component → One module

```
dmlite/
    manifests/          # puppet dsl
    templates/          # ruby erb
    tests/
    Modulefile          # for puppetforge
    Readme                 #
```

Manifests in a module

```
manifests/
    init.pp              # class{“dmlite”:}
    config.pp            # class{“dmlite::config”:}
    install.pp
    service.pp
    gridftp.pp
    srm.pp
    plugins/
        mysql.pp         # class{“dmlite::plugins::mysql”}
        mysql/
```

Puppet DSL Strengths

```
service{“httpd”:
    ensure => running
}
```

- Declarative
- Understands useful concepts

Puppet DSL Strengths /2

```
file{“/etc/yum.conf”:
    ensure => file,
    content =>template(“yum.conf.erb”),
    seltype = “http_sys_content_t”
}
```

- Its own templates
- Understands selinux

Puppet DSL Strengths /3

```
class lcgdm::ns::config(
    db_user = hiera(“dbuser”, “dpmmgr”)
) { … }
```

- Supports database input
- Encrypted storage of secrets

First Conclusion

- The Puppet DSL is great for configuration
- Many modules are already there
  on github, patches are welcome
- Easy to install modules

- Complete system configuration

Some not so obvious differences

- Some things are hard in puppet DSL
  - No for loops
  - Adjusting config files
  - Concurrency
  - Compatibility between modules
  - Boolean values and nil
- Suddenly you have more infrastructure to manage

Adjusting config files

- Use puppet templates
  - Full customized
  - Overwrites anything existing
- Exec{“sed bla /etc/yum.conf“}
  - Yaim comfort
- Augeas module
  - Respects existing changes
  - Quite complicated

Concurrency

- There's NO order in your manifest
- Almost all dependencies must be explicit

```
Class[Mysql::Server] -> Class[Lcgdm::Dpm::Service]
Class[Dmlite::Plugin::Mysql] ~> Class[Dmlite::Dav]
```

Compatibility between modules

- Resources can only be declared once globally

```
class lcgdm{
  user{“dpmmgr”}
}
class mysql( $user ) {
  user{$user}
}

class{“lcgdm”}
class{“mysql”: user => ‘dpmmgr’}  => BREAKS
```

Compatibility between modules /2

- Use the stdlib:

```
class mysql( $user ) {
 ensure_resource(user,$user,$user_params)
}
```

- Many more great functions!

Boolean values and nil

```
class{“lcgdm::ns::config”:
    dbmanage => “false”
}
```

- Every string is true

dbmanage => false

- stdlib functions: str2bool, num2bool, validate_bool
- Puppet DSL **undef** is an empty string in ruby

Infrastructure management

- Puppetmaster must be available
- Modules must be developed and deployed

- Is its own infrastructure project on a large scale

- PTs don't have to care ;)

Some module guidelines

- Split modules

```
class “lcgdm::ns” {
        Class[“Lcgdm::Ns::Install] -> Class[“Lcgdm::Ns::Config]
        Class[“Lcgdm::Ns::Config] ~> Class[“Lcgdm::Ns::Service]

        class{“lcgdm::ns::install”:}
        class{“lcgdm::ns::config”:}
        class{“lcgdm::ns::service”:}
}
```

- But …

Some module guidelines /2

- Make them configurable from the top class

```
class{“lcgdm::ns”:
        … parameters …
}
```

Manifests

- Need explicit dependencies
- Variable format different than YAIM
  - Real data structures now ;)

Manifests example

- yaim:

```
DPM_XROOTD_FEDREDIRS="atlas-xrd-uk.cern.ch:1094:1098,atlas,/
                                                    atlas”
```

- puppet:

```
$atlas_fed = {
name           => 'atlas',
fed_host       => 'atlas-xrd-uk.cern.ch',
xrootd_port    => 1094,
cmsd_port      => 1098,
local_port     => 11000,
namelib_prefix => "/dpm/cern.ch/home/atlas",
namelib        => "libXrdOucName2NameLFC.so root=/dpm/
   cern.ch/home/atlas match=dpmhost.example.com",
setenv         => $atlas_env,
paths          => [ '/atlas', '/atlas/scratch' ]
 }
```

Summary

- Most modules are easy to write
  - Some require serious thinking
- There are hundreds of examples
- Hacks are discouraged

- Sysadmins should understand puppet
  - Manifests are not always straight-forward

Puppetforge

- It's easy.
  And quick.

- Is it reliable?
  We don't know.

Puppet infrastructure

- Large CERN effort
  - Agile infrastructure
- Gets complicated on large scale
  - Multiple masters, failover and load-balancing, many configurations
- Probably worth if you manage a computer centre
- Talk to CERN agile infrastructure people ☺

Things we have

- Modules to look at:
  - Easy: dmlite
    puppet module install lcgdm-dmlite
  - Complicated: xrootd
    puppet module install lcgdm-xrootd
- Example manifests:
  https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm/Admin/Installation_Configuration_PuppetYaim2puppet
- helper script:
  https://svnweb.cern.ch/trac/lcgdm/wiki/YAIM2Puppet

Things we have /2

- Git repositories
  - CERN git it-puppet-module-<modulename>
    e.g. it-puppet-module-dmlite
  - Access:
    http://information-technology.web.cern.ch/services/git-service

Thank you!

Questions?