libral

a systems management API for Linux

David Lutterkort
@lutterkort
lutter@puppet.com
Augeas - a configuration API

Augeas is a configuration editing tool. It parses configuration files in their native formats and transforms them into a tree. Configuration changes are made by manipulating this tree and saving it back into native config files.

Augeas is:

- An API provided by a C library
- A command line tool to manipulate configuration from the shell (and shell scripts)
- Language bindings to do the same from your favorite scripting language
- Canonical tree representations of common configuration files
- A domain-specific language to describe configuration file formats

Augeas goals:

- Manipulate configuration files safely, safer than the ad-hoc techniques generally used with grep, sed, awk and similar mechanisms in scripting languages
- Provide a local configuration API for Linux
- Make it easy to integrate new config files into the Augeas tree

Take the introductory tour to explore the current implementation in more detail.

http://augeas.net/
The trouble with management
usermod -s /sbin/nologin app
$ usermod -s /sbin/nologin app
usermod: user 'app' does not exist
usermod -s /sbin/nologin app
usermod: user 'app' does not exist

grep -q app /etc/passwd && \
usermod -s /sbin/nologin app || \
useradd -u 72 -g 72 -s /sbin/nologin \
-M -r -d / app
Need to do this for every sort of resource

(user, group, package, service, firewall rule, …)
What if all you need is some insight?
Insight is its own use case

- regular audits
- verify that what you built is what you meant
- inspect running containers
- ask adhoc questions of your infrastructure
$ grep app /etc/passwd | \ncut -d ':' -f 7
/sbin/nologin
```bash
$ grep app /etc/passwd | \n  cut -d ':' -f 7

/bin/bash
/sbin/nologin
/sbin/nologin
```
Adhoc scripting full of pitfalls
Is "learn this big config management system"
the best we can do?
Anatomy of a configuration management system
What makes a good management API?
Desired state
(idempotency)
Bidirectional

(change & insight)
Light-weight abstractions
Simple to extend
But what about ... ?
What about CIM ?
What about OpenLMI?
What about $cm\_system$?
Writing providers
1. Pick scripting language and calling convention
2. Write standalone script
   a. Start with listing resources
   b. Get update working
3. Try out/integrate with ralsh
API complexity vs power

Power

simple  json  ???  C++

Complexity
Choosing a language

- Use what you are familiar with
- Keep it simple
  - bash
  - whatever can be expected to be there
  - mRuby
High-level API

\[ \text{provider.get(names)} \rightarrow \text{list[resource]} \]

\[ \text{provider.set(is, should)} \rightarrow \text{list[change]} \]
# 'Simple' calling convention

```sh
$ systemctl.prov ral_action=describe
  ↘ some yaml

$ systemctl.prov ral_action=list
  ↘ all resources

$ systemctl.prov ral_action=find name=<name>
  ↘ one resource
```
# 'Simple' calling convention

```bash
$ systemd.prov ral_action=update \ 
   attr1=value1 \ 
   attr2=value2

changes performed
```
---

**provider:**

- **type:** service
- **invoke:** simple
- **actions:** [list, find, update]
- **suitable:** `${suitable}`
- **attributes:**
  - **name:**
    - **type:** string
  - **ensure:**
    - **type:** enum[running, stopped]
  - **enable:**
    - **type:** enum[true, false, mask]
provider:
  type: service
  invoke: simple
  actions: [list, find, update]
  suitable: ${suitable}
attributes:
  name:
    type: string
  ensure:
    type: enum[running, stopped]
  enable:
    type: enum[true, false, mask]
# systemd provider: 'list' all known services

```bash
list() {
    echo '# simple'
    join -j 1 -o 1.1,1.2,2.3 \
        <(systemctl list-unit-files ...) \
        <(systemctl list-units --type service ...) \
    | while read svc enable ensure \
    do \
        echo "name: $svc"
        echo "ensure: $ensure"
        echo "enable: $enable"
    done
}
```
# systemd provider: 'list' all known services
list() {
    echo '# simple'
    join -j 1 -o 1.1,1.2,2.3 \\n        <(systemctl list-unit-files ...) \\n        <(systemctl list-units --type service ...) \\n    | while read svc enable ensure \
    do \
        echo "name: $svc"
        echo "ensure: $ensure"
        echo "enable: $enable"
    done
}
# systemd provider: 'list' all known services

```bash
list() {
    echo '# simple'
    join -j 1 -o 1.1,1.2,2.3 \
        <(systemctl list-unit-files ...) \
        <(systemctl list-units --type service ...) \
    | while read svc enable ensure \
     do \
        echo "name: $svc"
        echo "ensure: $ensure"
        echo "enable: $enable"
     done
}
```
What now?
Firm up interfaces/calling conventions
More providers
Distribution
Desired-state bidirectional API

join the fun
https://github.com/puppetlabs/libral
puppet

The shortest path to better software.