puppet

libral

a systems management API for Linux

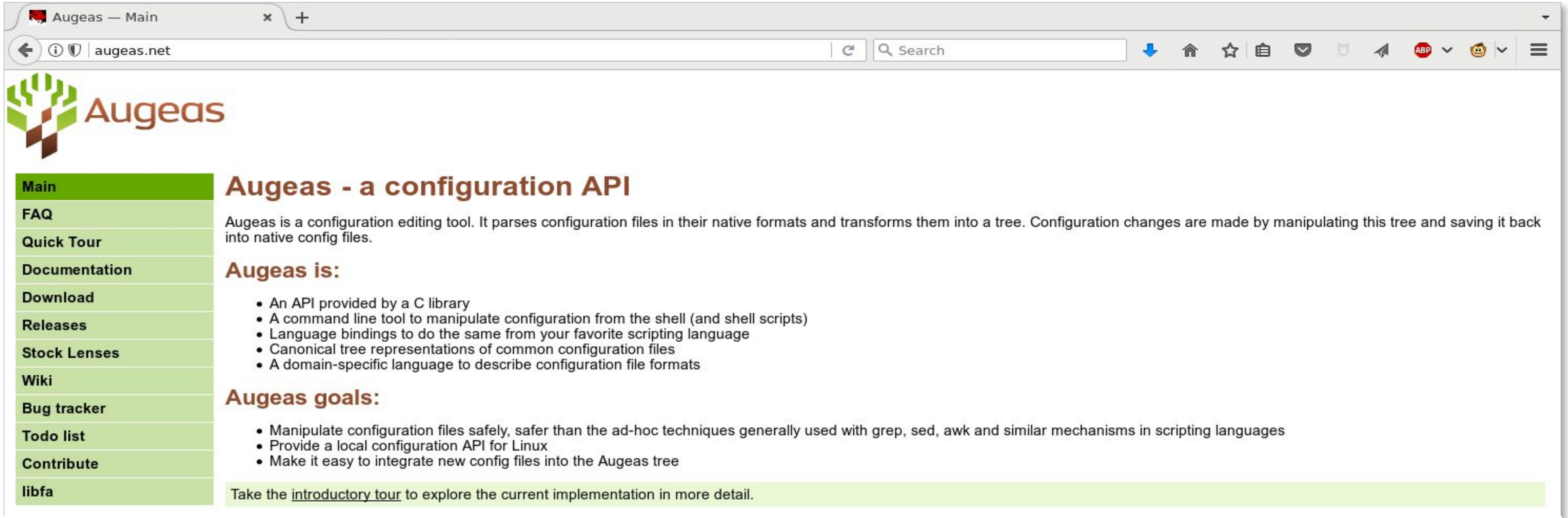
David Lutterkort

@lutterkort

lutter@puppet.com

puppet

The shortest path
to better software.



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

puppet

```
$ 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 | \
    cut -d ':' -f 7
/sbin/nologin
```

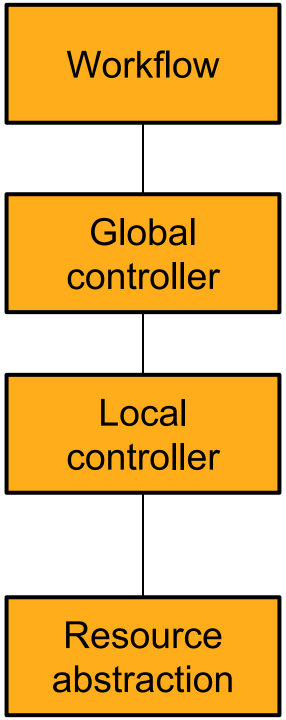
```
$ grep app /etc/passwd | \
    cut -d ':' -f 7
/bin/bash
/sbin/nologin
/sbin/nologin
```

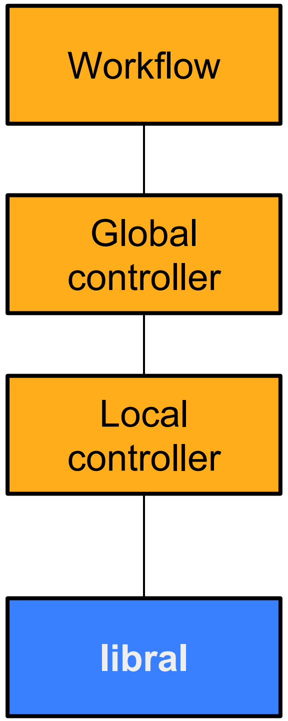
Adhoc scripting full of pitfalls

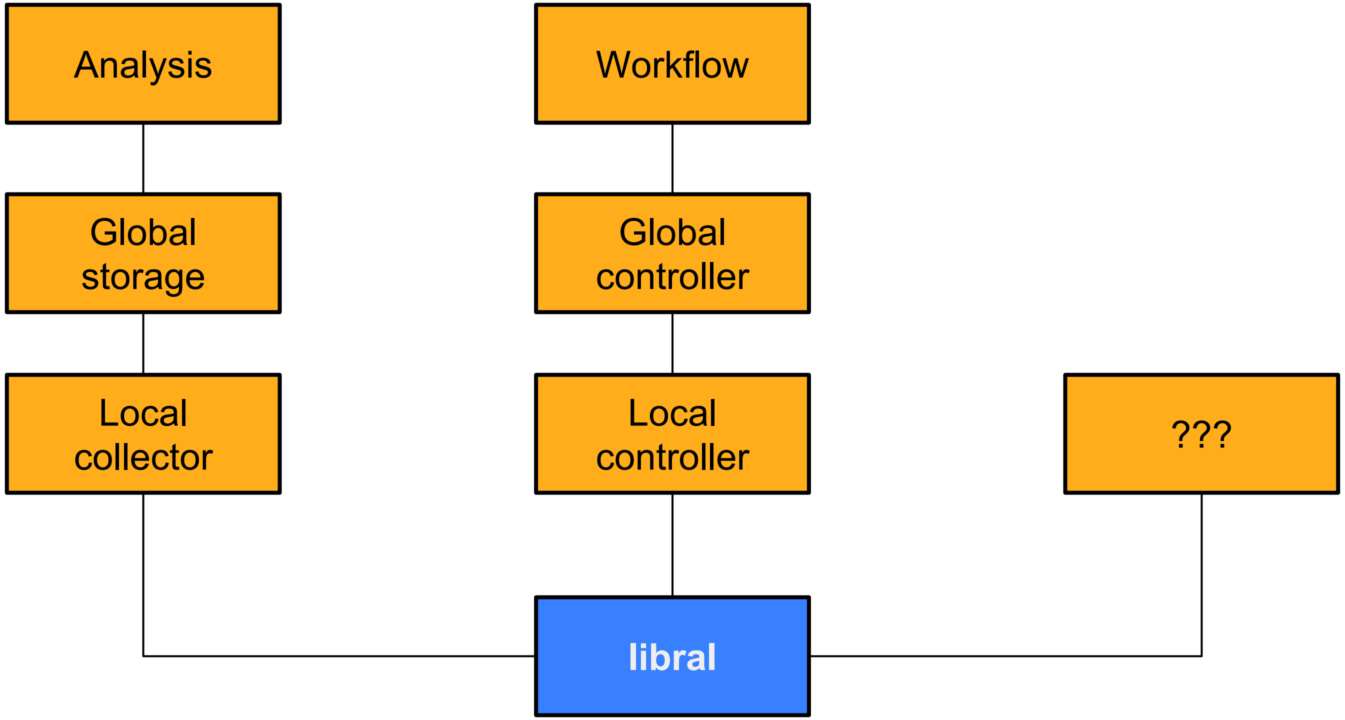
Is "learn this big config management system" the best we can do ?

Anatomy of a configuration management system

puppet


Workflow
Global controller
Local controller
Resource abstraction


Workflow
Global controller
Local controller
libral


Analysis
Workflow
Global storage
Global controller
Local collector
Local controller
???
libral

What makes a good management API ?

puppet

Desired state

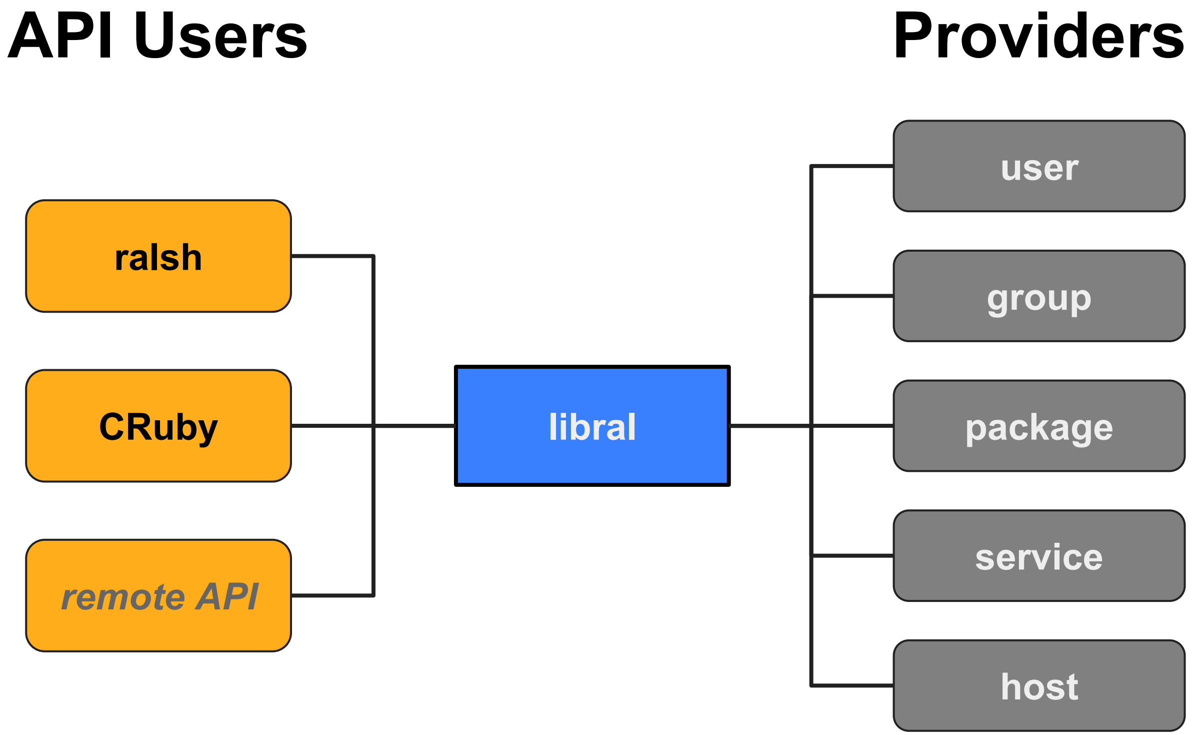
(idempotency)

Bidirectional

(change & insight)

Light-weight abstractions

Simple to extend


API Users
Providers
ralsh
CRuby
remote API
libral
user
group
package
service
host

But what about … ?

puppet

What about CIM ?

What about OpenLMI ?

What about `$cm_system` ?

Demo

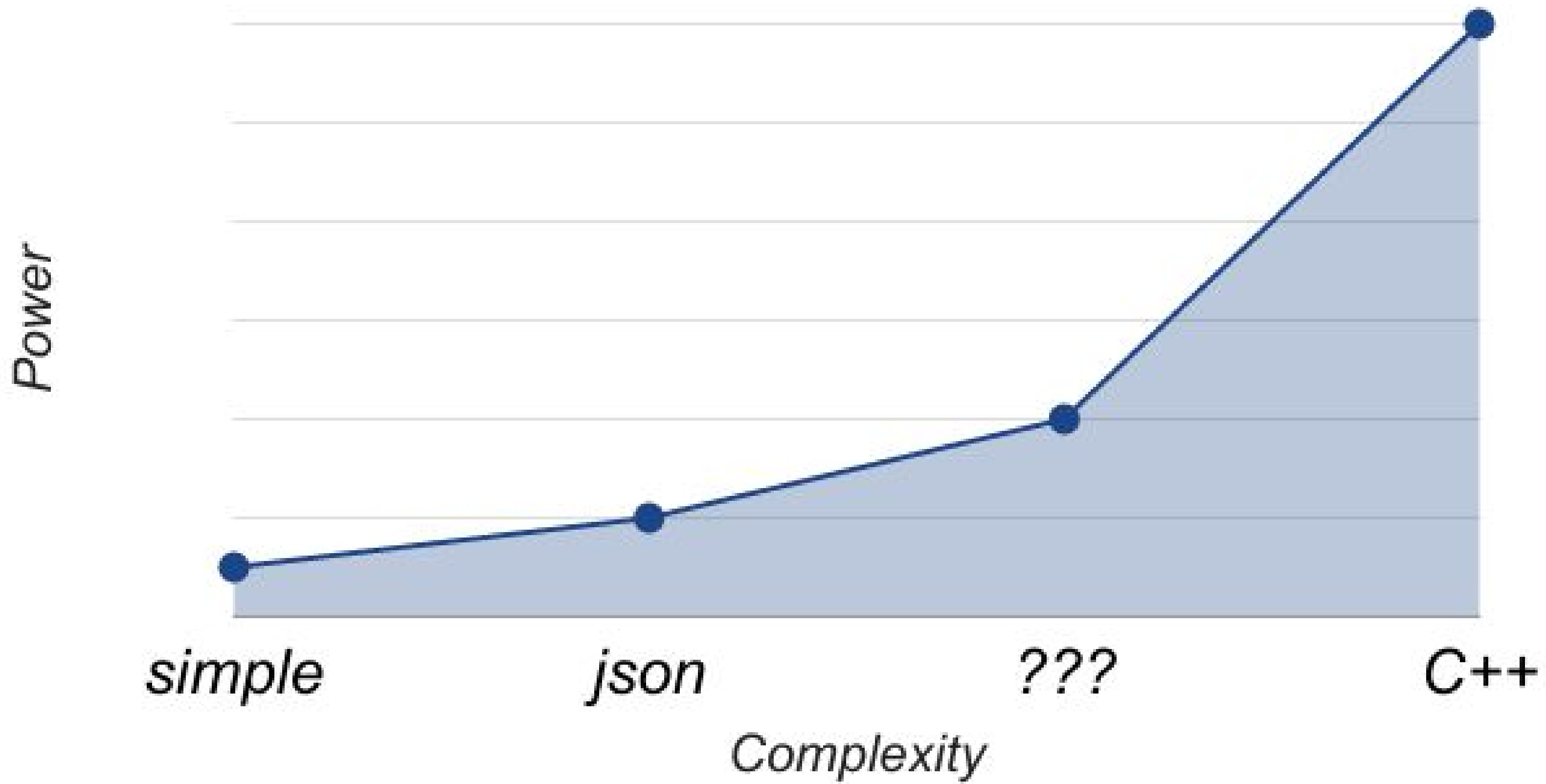
puppet

Writing providers

puppet

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



Choosing a language

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

High-level API

```
provider.get(names) → list[resource]

provider.set(is, should) → list[change]
```

```
# 'Simple' calling convention

$ systemd.prov ral_action=describe
  ↳ some yaml

$ systemd.prov ral_action=list
  ↳ all resources

$ systemd.prov ral_action=find name=<name>
  ↳ one resource
```

```
# 'Simple' calling convention

$ systemd.prov ral_action=update \
    attr1=value1                 \
    attr2=value2
  ↳ changes performed
```

```yaml
---
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]
```

```yaml
---
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
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 \
        <(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 \
        <(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

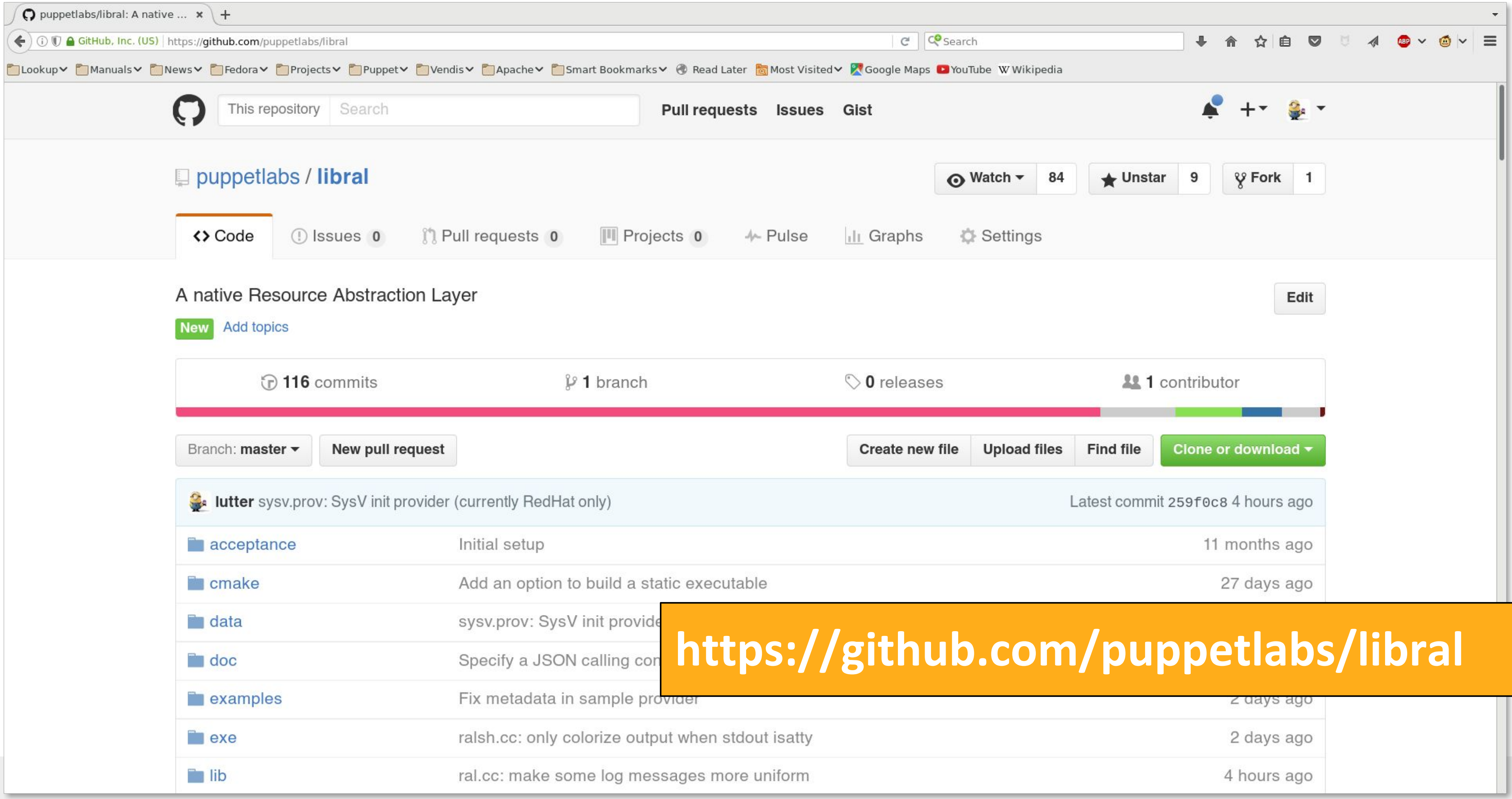
More providers

Distribution

Desired-state bidirectional API

join the fun

puppet



puppetlabs / libral

Watch 84 · Unstar 9 · Fork 1

Code · Issues 0 · Pull requests 0 · Projects 0 · Pulse · Graphs · Settings

A native Resource Abstraction Layer

116 commits · 1 branch · 0 releases · 1 contributor

Branch: master · New pull request · Create new file · Upload files · Find file · Clone or download

lutter sysv.prov: SysV init provider (currently RedHat only) — Latest commit 259f0c8 4 hours ago

acceptance	Initial setup	11 months ago
cmake	Add an option to build a static executable	27 days ago
data	sysv.prov: SysV init provide	
doc	Specify a JSON calling con	
examples	Fix metadata in sample provider	2 days ago
exe	ralsh.cc: only colorize output when stdout isatty	2 days ago
lib	ral.cc: make some log messages more uniform	4 hours ago

https://github.com/puppetlabs/libral

puppet

The shortest path
to better software.