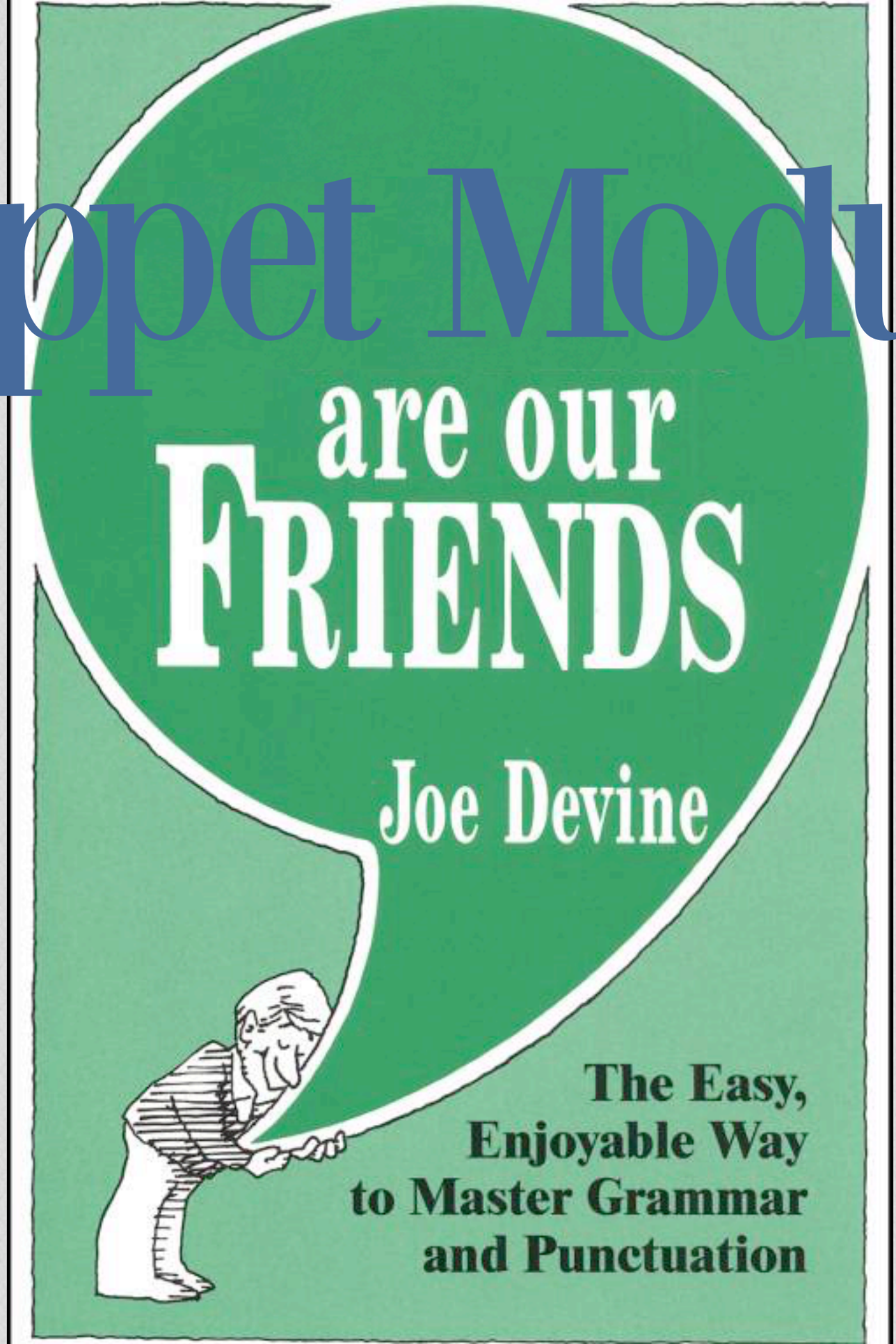
Puppet Modules



constructing a module

designing good modules

putting modules together

constructing a module

our mission

```
# TODO: Turn this pattern into a puppet module
file { "/etc/init/collect-cpu-stats.conf":
  content => template("etc/init/collect-cpu-stats.conf.erb")
}
file { "/etc/init.d/collect-cpu-stats":
  ensure  => "/lib/init/upstart-job",
}
remotefile { "/usr/local/bin/collect-cpu-stats":
  mode => 755,
  notify => Service["collect-cpu-stats"],
}
service { collect-cpu-stats:
  ensure  => running,
  require => [
    File["/usr/local/bin/collect-cpu-stats"],
    File["/etc/init/collect-cpu-stats.conf"],
    File["/etc/init.d/collect-cpu-stats"],
  ]
}
```

a place for everything...

```
$ tree puppet/modules/collect_cpu_stats

collect_cpu_stats                  # module root dir
├── files                          # static files
├── lib
│   ├── facter                     # custom facts
│   └── puppet
│       ├── parser
│       │   └── functions # custom parser functions
│       └── type                   # native puppet types
├── manifests
│   └── init.pp                    # *** entry point ***
└── templates                      # template files
```

...and everything in its place

```
$ tree puppet/modules/collect_cpu_stats

collect_cpu_stats
├── files
│   └── collect-cpu-stats
├── manifests
│   ├── config.pp
│   ├── init.pp
│   ├── script.pp
│   └── service.pp
└── templates
    └── collect-cpu-stats.conf.erb
```

the classic init.pp

manifests/init.pp

```
class collect_cpu_stats {
  include collect_cpu_stats::package
  include collect_cpu_stats::config
  include collect_cpu_stats::service

  Class['collect_cpu_stats::package']
    -> Class['collect_cpu_stats::config']
    -> Class['collect_cpu_stats::service']
    -> Class['collect_cpu_stats']
}
```

adding parameters

manifests/init.pp

```
class collect_cpu_stats (
  $hostname = $fqdn
  $interval = 10
) {
  include collect_cpu_stats::package
  class { 'collect_cpu_stats::config'
    hostname => $hostname,
    interval => $interval
  }
  include collect_cpu_stats::service

  Class['collect_cpu_stats::package']
    -> Class['collect_cpu_stats::config']
    -> Class['collect_cpu_stats::service']
    -> Class['collect_cpu_stats']
}
```

customize names for clarity

manifests/init.pp

```
class collect_cpu_stats (
  $hostname = $fqdn
  $interval = 10
) {
  include collect_cpu_stats::script
  class { 'collect_cpu_stats::config'
    hostname => $hostname,
    interval => $interval
  }
  include collect_cpu_stats::service

  Class['collect_cpu_stats::script']
    -> Class['collect_cpu_stats::config']
    -> Class['collect_cpu_stats::service']
    -> Class['collect_cpu_stats']
}
```

script.pp: sourcing files

manifests/script.pp

```
class collect_cpu_stats::script {
  file { '/usr/local/bin/collect-cpu-stats':
    mode => 755,
    source => 'puppet:///modules/collect_cpu_stats/collect-cpu-stats'
  }

  file { '/etc/init.d/collect-cpu-stats':
    ensure => link,
    target => '/lib/init/upstart-job'
  }
}
```

config.pp: using templates

manifests/config.pp

```
class collect_cpu_stats::config (
  $hostname,
  $interval
) {
  file { '/etc/init/collect-cpu-stats.conf':
    content => template('collect_cpu_stats/collect-cpu-stats.conf.erb'),
    notify => Class['collect_cpu_stats::service']
  }
}
```

service.pp: simple as it gets

manifests/service.pp

```
class collect_cpu_stats::service {
  service { 'collect-cpu-stats':
    ensure  => running,
  }
}
```

ta da!

```
$ tree puppet/modules/collect_cpu_stats

collect_cpu_stats
├── files
│   └── collect-cpu-stats
├── manifests
│   ├── config.pp
│   ├── init.pp
│   ├── script.pp
│   └── service.pp
└── templates
    └── collect-cpu-stats.conf.erb
```

designing good modules

it's just unix philosophy

Rule of Modularity
Rule of Clarity
Rule of Composition
Rule of Separation
... etc.

unix philosophy in puppet

"do one thing well"

require/notify classes, not types

design for composability

pull configuration up to init.pp

intentional interface design

use the puppet style guide

one class/define per file

one parameter per line

quoting, spacing, etc.

when in doubt, check it!

putting modules together

SysAdvent

DECEMBER 13, 2012

Day 13 - Configuration Management as Legos

This was written by Adrien Thebo.

Configuration management is hard. Configuring systems properly is a lot of hard work, and trying to manage services and automate system configuration is a serious undertaking.

Even when you've managed to get your infrastructure organized in Puppet manifests or Chef cookbooks, organizing your code can get ugly, fast. All too often a new tool has to be managed under a short deadline, so any sort of code written to manage it solves the immediate problem and no more. Quick fixes and temporary code can build up, and before you know it, your configuration management becomes a tangled mess. Nobody intends for their configuration management tool to get out of hand, but without guidelines for development, all it takes is a few instances of `git commit -a -m 'Good enough'` for the rot to set in.

Organizing configuration management code is clearly a good idea, but how do you do it? For normal development, there are many of design patterns for laying out and organizing programs and libraries. Traditional software development has had around 40 years to mature, and config management is fairly young by comparison and hasn't had the time to have formal best practices.

types of puppet modules

“base-blocks”: generic to the world

“weird-blocks”: specific to us

“site-services”: assemble blocks

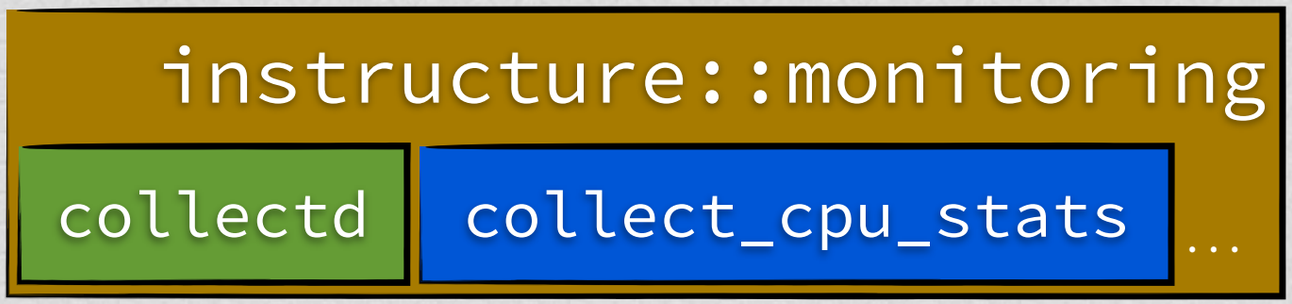
“site-roles”: assemble services

build it up

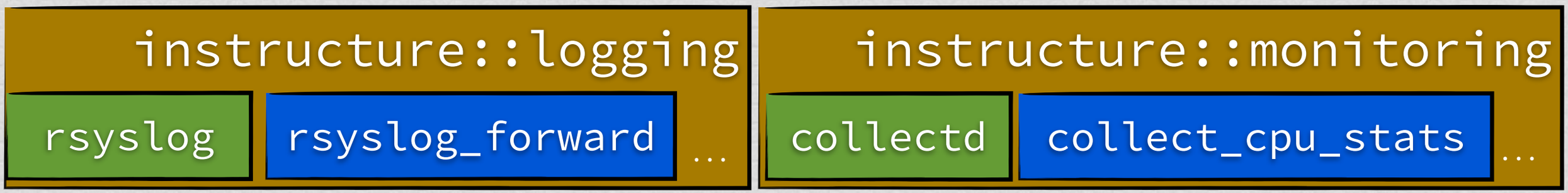
collectd

build it up

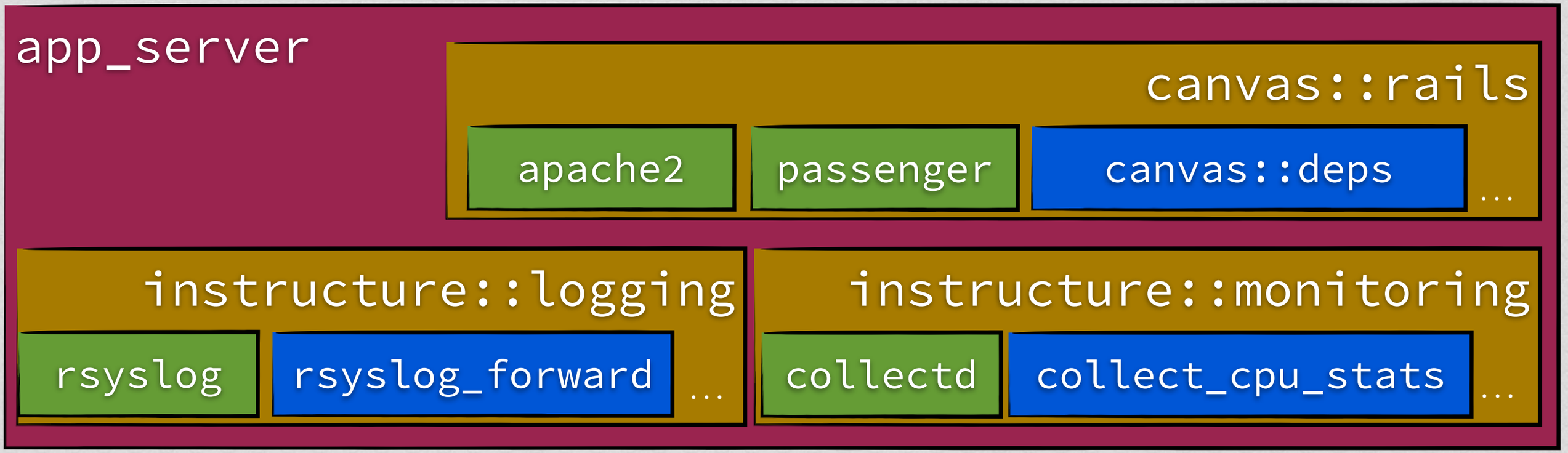
build it up



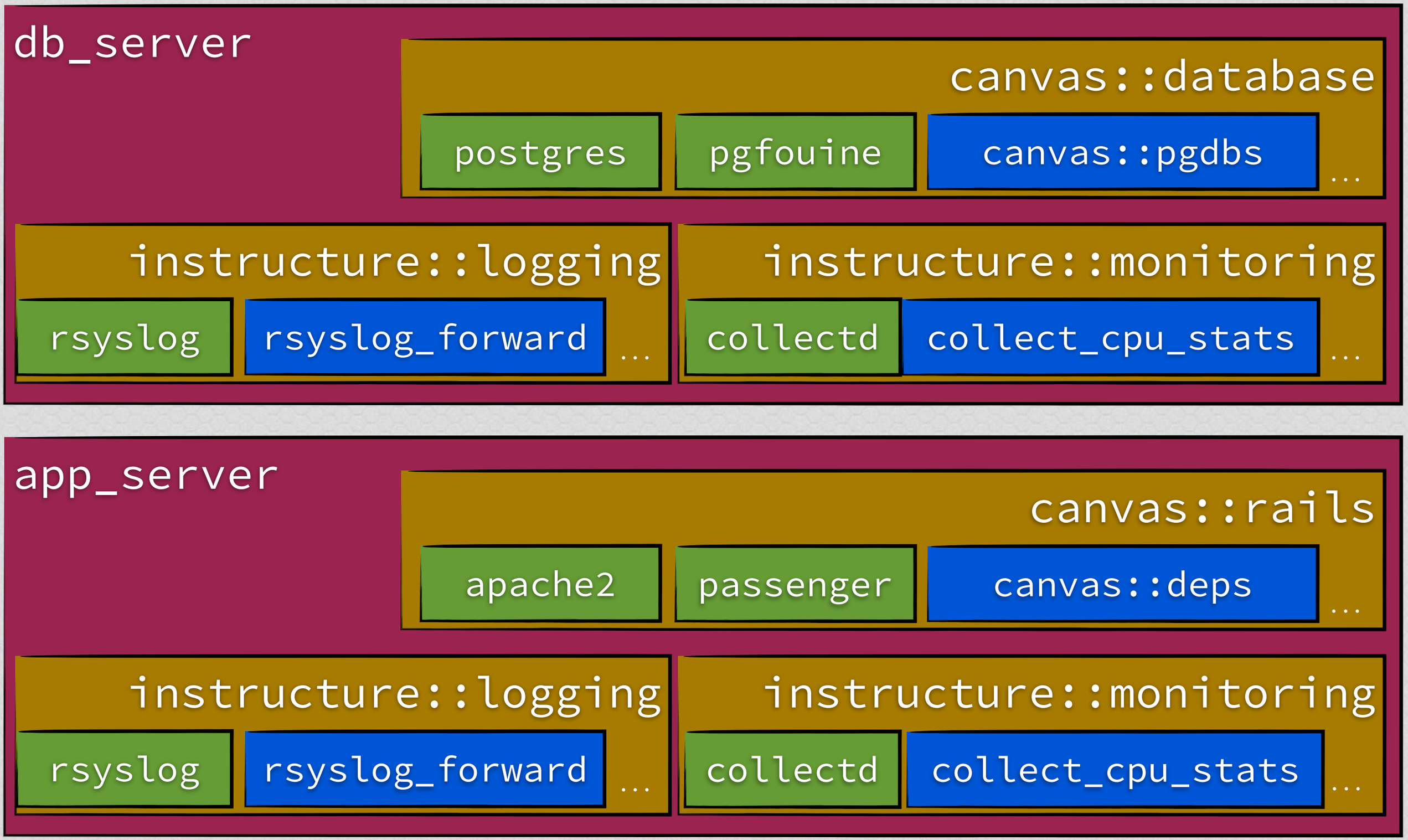
build it up


instructure::logging
rsyslog
rsyslog_forward
...
instructure::monitoring
collectd
collect_cpu_stats
...

build it up


app_server
canvas::rails
apache2
passenger
canvas::deps
...
instructure::logging
rsyslog
rsyslog_forward
...
instructure::monitoring
collectd
collect_cpu_stats
...

build it up


db_server
canvas::database
postgres
pgfouine
canvas::pgdbs
...
instructure::logging
rsyslog
rsyslog_forward
...
instructure::monitoring
collectd
collect_cpu_stats
...
app_server
canvas::rails
apache2
passenger
canvas::deps
...
instructure::logging
rsyslog
rsyslog_forward
...
instructure::monitoring
collectd
collect_cpu_stats
...

Puppet Modules