Everything as a Cookbook

service-oriented thinking for your code

CHEF™
CODE CAN

Who is this guy?

Tom Duffield
Consulting Engineer with Chef

tom@getchef.com

@tomduffield

tduffield

tomduffield.com

Good Practices

- Everyone wants “best practices”…until they don’t.
- These are “good practices,” based on good development practices and what I have seen work over my numerous years of Chef cookbook development.

There are many established patterns already…

LWRPS

Application Wrapper Cookbook

HWRPs

Environment Cookbook

Cookbook

Wrapper Cookbook

Library Cookbook

Role Cookbook

…but developers still have a lot of questions

- When should I use wrapper cookbooks?
- Should a cookbook use node attributes, LWRPs or both to modify behavior?
- Should I write a custom resources as LWRPs or HWRPs?
- Is it still a Library cookbook even though there is more than just Libraries in it?

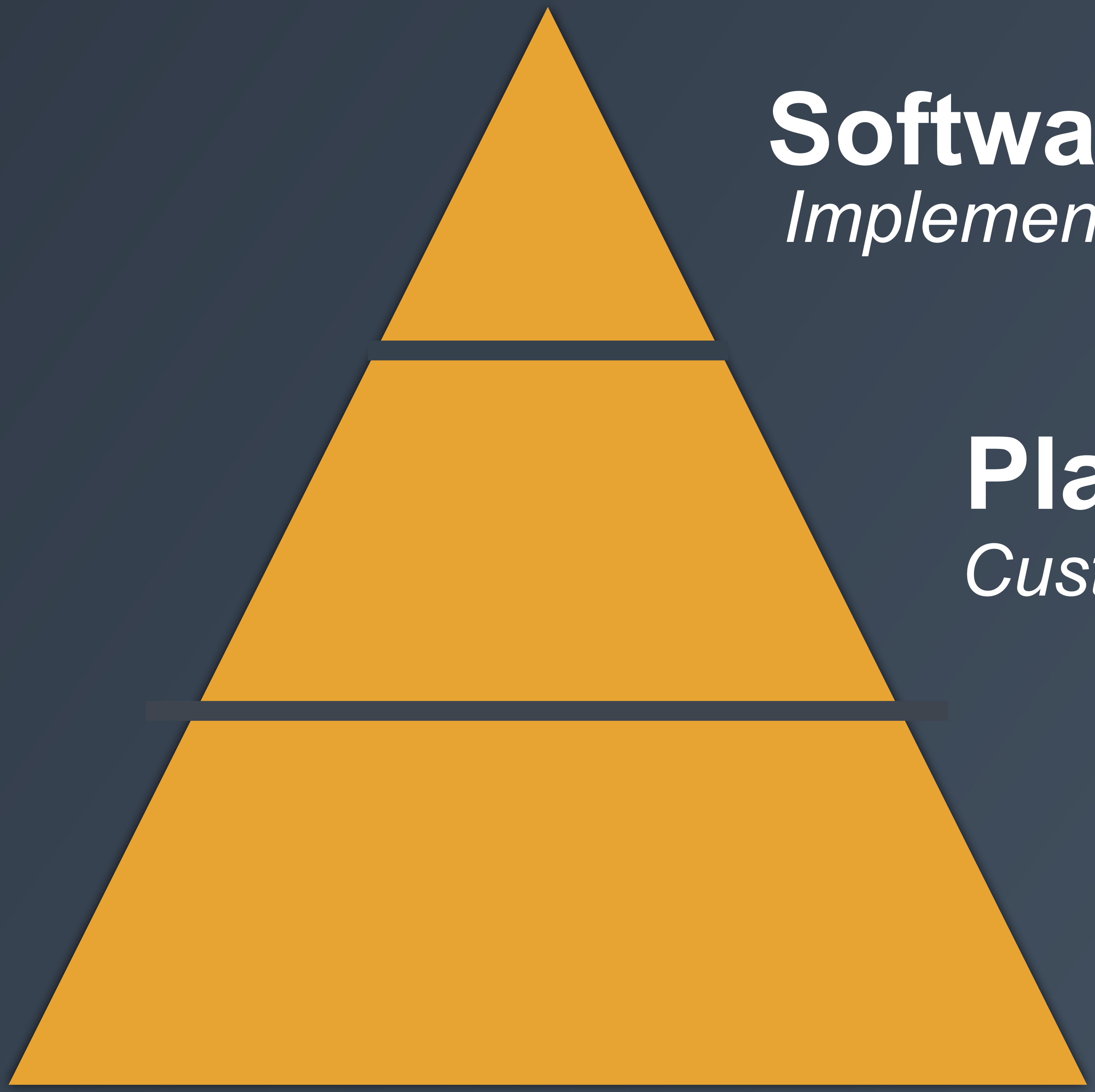
The Problem

There are clear guidelines on what patterns to use or when to use them.

Solution

Create reusable patterns based on what you are trying to automate, not how.

Start by referencing a well established model

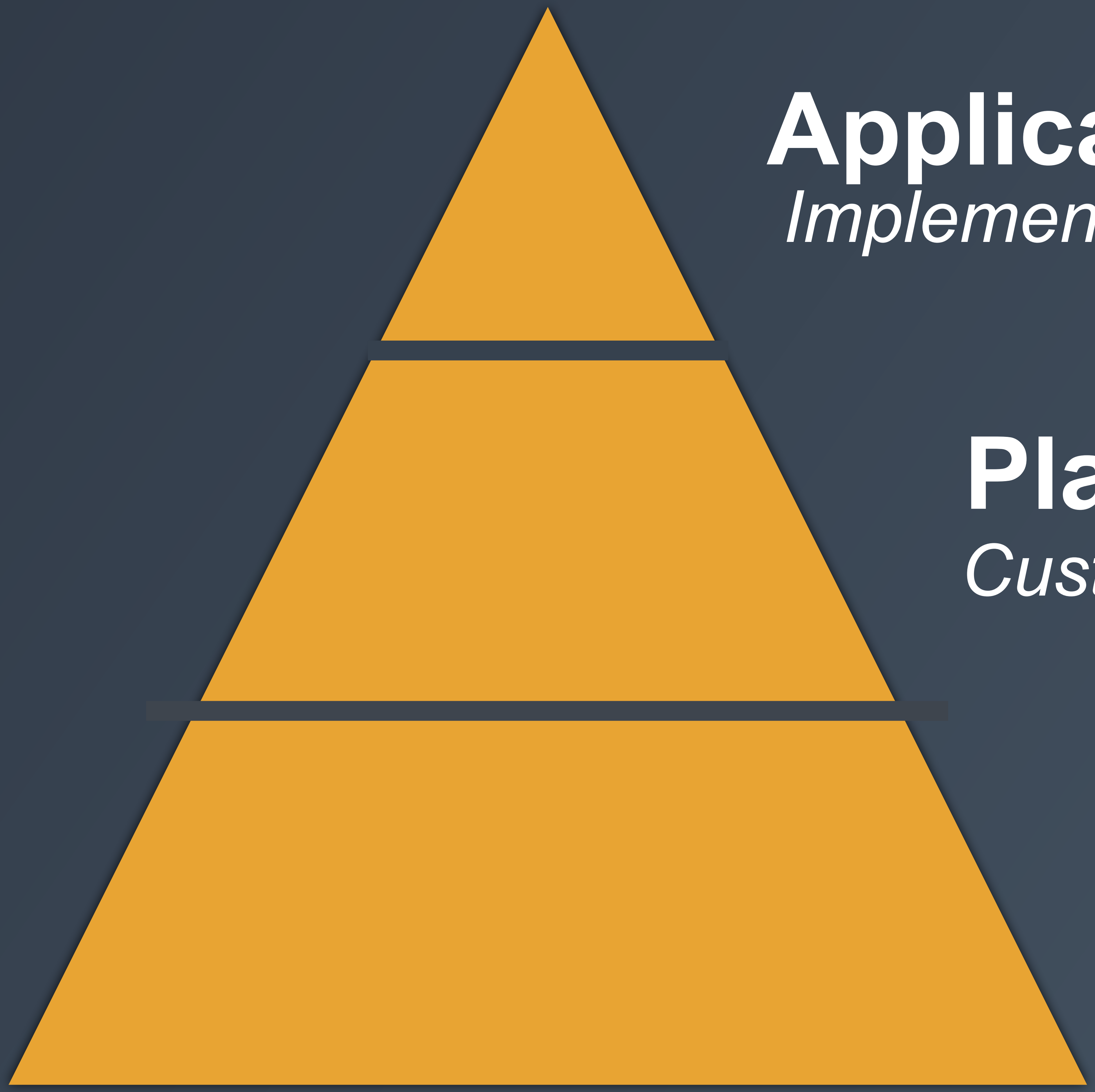


Software as a Service
Implementation of a platform that delivers a working application.

Platform as a Service
Customizable execution environment.

Infrastructure as a Service
Basic building blocks for computing.

⌘+C, ⌘+V



Application (as a) Cookbook
Implementation of a platform that delivers a working application.

Platform (as a) Cookbook
Customizable execution environment.

Infrastructure (as a) Cookbook
Basic building blocks for computing.

Global Patterns

Global Patterns

- Recipes should be modular to allow users to be selective about the policies they enforce.

```
🗁 chef-client
    ├──── recipes
    │       ├──── config.rb
    │       ├──── cron.rb
    │       ├──── default.rb
    │       ├──── delete_validation.rb
    │       ├──── service.rb
    │       ├──── task.rb
```

Global Patterns

- Recipes should minimally prescriptive. Don’t force people to subscribe to unnecessary patterns.

cron/recipes/default.rb

```ruby
package 'cron' do
  package_name case node['platform_family']
               when 'rhel', 'fedora'
                 node['platform_version'].to_f >= 6.0 ? 'cronie' : 'vixie-cron'
               when 'solaris2'
                 'core-os'
               end
end

service 'cron' do
  service_name 'crond' if platform_family?('rhel', 'fedora')
  service_name 'vixie-cron' if platform_family?('gentoo')
  action [:enable, :start]
end
```

Global Patterns

- Repeatable patterns for implementation-specific configuration should be exposed as custom resources instead of recipes.

my_site/recipes/web_server.rb

```
# Exposed by the apache2 cookbook
web_app "my_site" do
  server_name       node['hostname']
  server_aliases    [node['fqdn'], "my-site.example.com"]
  cookbook          "my-site"
  template          "my-custom-vhost.conf.erb"
  docroot           "/srv/www/my_site"
end

web_app "my_site_french" do
  server_name       node['hostname']
  server_aliases    [node['fqdn'], "fr.my-site.example.com"]
  cookbook          "my-site"
  template          "my-custom-fr-vhost.conf.erb"
  docroot           "/srv/www/my_site_fr"
end
```

Infrastructure Cookbooks

Infrastructure Cookbooks

- Manage the basic building blocks of your nodes:
    - Operating System - package managers like yum, core services like ntp and cron, etc.
    - Storage - LVM, RAID, etc.
    - Networking - hosts files, DNS, firewalls, route tables, etc.
    - Programming Languages - php, perl, ruby, java, etc.
    - Development Utilites - Chef runtime libs (chef-sugar), system libs (make, gcc)

Infrastructure Cookbooks

- Cookbooks have no (or very few) dependencies on other cookbooks.

apt/metadata.rb

```ruby
name              'apt'
maintainer        'Chef Software, Inc.'
maintainer_email  'cookbooks@opscode.com'
license           'Apache 2.0'
description       'Configures apt and apt services'
long_description  IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version           '2.3.9'

%w{ ubuntu debian }.each do |os|
  supports os
end
```

Platform Cookbooks

Platform Cookbooks

- Manage execution environments:
  - Execution Runtime - tomcat, nodejs, rails, etc.
  - Web Server - apache2, nginx, etc.
  - Database - mysql, postgresql, riak, cassandra, etc.
  - Monitoring - sensu, zabbix, nagios, etc.

Platform Cookbooks

- Typically depends on a handful of infrastructure cookbooks.

nginx/metadata.rb

```ruby
name              'nginx'
maintainer        'Opscode, Inc.'
maintainer_email  'cookbooks@opscode.com'
license           'Apache 2.0'
description       'Installs and configures nginx'
version           '2.6.3'

depends 'apt',             '~> 2.2'
depends 'bluepill',        '~> 2.3'
depends 'build-essential', '~> 2.0'
depends 'ohai',            '~> 1.1'
depends 'runit',           '~> 1.2'
depends 'yum-epel',        '~> 0.3'
```

Platform Cookbooks

- Recipes utilize a combination of core Chef and custom resources exposed by infrastructure and other platform cookbooks.

nginx/recipes/repo.rb

```
include_recipe 'apt::default'

apt_repository 'nginx' do
  uri          node['nginx']['upstream_repository']
  distribution node['lsb']['codename']
  components   %w(nginx)
  deb_src      true
  key          'http://nginx.org/keys/nginx_signing.key'
end
```

Application Cookbooks

Application Cookbooks

- Implementation of a platform that delivers your application.
  - Associated one-to-one with an application - website, api, etc.
  - There are only few good examples on the web because these are not really designed to be shared.

    tduffield/myface

Application Cookbooks

- Modularity (again)! Application modules or tiers are broken up into separate recipes.

```
🗁 myface
   ├──── metadata.rb
   ├──── recipes
   │      ├──── database.rb
   │      ├──── default.rb
   │      └──── webserver.rb
```

Application Cookbooks

- The default recipe should install a full development stack.

myface/recipes/default.rb

```
include_recipe 'myface::database'
include_recipe 'myface::webserver'
```

Application Cookbooks

- Leverages core Chef resources as well as custom resources exposed by Infrastructure and Platform cookbooks.

myface/recipes/database.rb

```
mysql_database node['myface']['database']['dbname'] do
  connection mysql_connection_info
end

cookbook_file node['myface']['database']['seed_file'] do
  source "myface-init.sql"
  owner "root"
  group "root"
end

execute "initialize myface database" do
  command my_mysql_initiate_command
  not_if  database_exists?
end
```

Application Cookbooks

- Instead of defining a `run_list` in your role, you essentially define your `run_list` by using a series of `include_recipe` statements and Chef resources.

myface/recipes/ci_server.rb

```
include_recipe "jenkins::java"
include_recipe "jenkins::master"

%w{
  git-client
  token-macro
  git
  github
  github-api
  ghprb
}.each do |plugin|
  jenkins_plugin plugin
end
```

Application Cookbooks & Roles

- Some people will replace roles entirely with Application Cookbook recipes.

```
role[myface_ci_server]
```

Application Cookbooks & Roles

- Some people will replace roles entirely with Application Cookbook recipes.

~~role[myface_ci_server]~~

```
recipe[myface::ci_server]
```

Application Cookbooks & Roles

- Others will simply map each Application Cookbook recipe 1:1 with a role.

roles/widget_web_server.json

```
{
  "name": "myface_ci_server",
  "run_list": [ "myface::ci_server" ]
}
```

Application Cookbooks & Roles

- Others will simply map each Application Cookbook recipe 1:1 with a role.

roles/widget_web_server.json

```
{
  "name": "myface_ci_server",
  "run_list": [ "myface::ci_server" ]
}
```

- Which one should you choose?
  - **Whichever you feel most comfortable with!**
  - I have used both in production before.

Application Cookbooks

- Control cookbook version constraints of your application in your metadata.rb

myface/metadata.rb

```ruby
name             'myface'
maintainer       'Tom Zuckerberg'
maintainer_email 'tom.zuckerberg@myface.co.uk'
license          'Apache 2.0'
description      'Installs/Configures myface'
long_description IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version          '2.0.0'

depends 'apache2', '~> 1.8.0'
depends 'mysql', '~> 4.0.0'
depends 'database', '~> 1.6.0'
depends 'php', '~> 1.3.0'
```

Application Cookbooks & Environments

- Control the cookbook constraints of your Application Cookbooks in your environment files.

environments/production.json

```
{
  "name": "production",
  "cookbook_versions": {
    "myface": "1.0.0",
    "theirface": "0.8.0"
  }
}
```

Application Cookbooks & Environments

- This pattern allows you to have multiple versions of a dependent cookbook in the same environment without issues.

```
{
  "name": "production",
  "cookbook_versions": {
    "apache": "4.0.0",
    "mysql": "5.0.0"
  }
}
```

Everyone in production will need to use apache v4.0.0

But here, `myface` could use apache v4.0 and `theirface` could use v3.8.

```
{
  "name": "production",
  "cookbook_versions": {
    "myface": "= 1.0.0",
    "theirface": "= 0.8.0"
  }
}
```

In Summary

- Recipes should be modular to allow users to be selective about the policies they enforce.
- Recipes should minimally prescriptive. Don’t force people to subscribe to unnecessary patterns.
- Repeatable patterns for implementation-specific configuration should be exposed as custom resources instead of recipes.
- Classify your cookbooks as Infrastructure, Platform and Application to
  - Help visualize the interactions between your different cookbooks.
  - Help keep your cookbooks modular and reusable.

Thank You

Office Hours

- Today at 2:05 to 2:20 in Marina (right after this talk)
- Otherwise, reach out to me online:

tom@getchef.com

@tomduffield

tduffield

tomduffield.com

Questions?

tom@getchef.com @tomduffield tomduffield.com