# Learn us some chef!

- Questions
- Concepts
- Execution workflow
- Development workflow
- Won't be covering advanced topics such as databags, LWRPs, more in-depth inheritence

# Questions

- What else should we cover?

# Header stuff

- Caveat
  - Not an authority
  - DIY or @techops on $CHATlemail
- google: `your_query + 'opscode'`

# Overview

- What is it?
  - A framework (~/code/chef) and set of tools (chef-client, knife, etc) to support configuration management
- What does it provide?
  - Repeatable builds defined in code

See other: puppet

# The Tao of Chef

- Nodes compose roles
- Roles compose recipes
- Cookbooks group recipes
- Templates are expanded with attributes
- Environments override attributes

*Run lists are the general mechanism for rolling up behaviors*

# Concepts

```
$ find ~/code/chef -type d -depth 1
```

- Nodes
- Roles
- Cookbooks
- Recipes
- Templates
- Attributes
- Environments

# Nodes

A (json) representation of a physical machine resource e.x. bouncer-production-01.json

Nodes have:

- Hostnames
- Node specific configs / overrides
- And a run list which specifies a list of roles

Moving away from nodes and towards just recipes and roles? I could be wrong

# Roles

Roles reference one or more recipes

- facebook_analytics_service_worker
  - → recipe[facebook-analytics-service::worker]
- facebook_analytics_service_webhead
  - → recipe[facebook-analytics-service::webhead]

# Roles

- facebook_analytics_service_standalone
  - → recipe[facebook-analytics-service::webhead]
  - → recipe[facebook-analytics-service::sched]
  - → ...

# Cookbooks

- Cookbooks group together related recipes, attributes and templates
- ex. facebook_analytics_service, message_optics_service

# Recipes

- Recipes use contextual data to expand templates
- Basically a bunch of ruby function calls to generate configs
- For example, 'create redis.conf with the timeout value provided in the cookbook's attributes def'

```
# cookbooks/twitter-analytics-service/recipes/redis.rb
socialcode_service_redis '/etc/redis/redis.conf' do
  action :create
  redis_server_timeout node[cookbook][:redis_server_timeout]
end
```

# Templates

- ERB files which are expanded by attributes and/or environment overrides
- nginx config, settings.py, etc
- For example, 'enable or disable DEBUG based on attributes'

```
# settings.py.erb
DEBUG = <%= @debug_mode %>
TEMPLATE_DEBUG = DEBUG
```

# Attributes

- Attributes define default template values
  - timeouts, port settings, db configs etc
- Will be overridden later if need be by things like environments
- Some parts not always obvious due to language mismatch

```
# default.rb
default[:project]['listeners'] = [[80, 'default']]
default[:project]['log_level'] = "WARN"
default[:project]['debug_mode'] = "False"
default[:project]['use_sentry'] = true
```

# Environments

- /environments/{alpha, staging, production}.json
- Environment override for stage (alpha, staging, prod)
- Helps with things like, 'fb analytics alpha should point to the alpha fb analytics database / cache cluster / bouncer environment / etc'

# The Tao of Chef

- Nodes compose roles
- Roles compose recipes
- Cookbooks group recipes
- Templates are expanded with attributes
- Environments override attributes

# Standalone example

How do the standalone hosts work (i.e. instagram-standalone-alpha-02.socialcodedev.com)?

- Create a role: facebook-analytics-standalone
- Set its run list to a standalone recipe

```
"run_list": ["recipe[facebook-analytics-service::standalone]"]
```

- In the recipe, include all project recipes

# Standalone example

```
$ cat cookbooks/facebook-analytics-service/recipes/standalone

include_recipe 'facebook-analytics-service::common'
include_recipe 'facebook-analytics-service::redis'
include_recipe 'facebook-analytics-service::sched'
include_recipe 'facebook-analytics-service::webhead'
include_recipe 'facebook-analytics-service::worker'
```

# Deployment: How thing happen

- Jenkins build gets triggered
- A new package is created as the output of the build and uploaded to our apt repo
- At the end of the build knife runs and triggers chef-client
- chef-client starts on the host(s) and
  - Installs packages
  - Installs build deb
  - Generates config files from templates
  - Installs the new build package
  - Stops / starts services
  - and so on

# Deployment

- Once an hour or so chef-client runs on our hosts and pulls the latest from the chef repo so that the latest merged in changes are available during deployments
- Services (generally) aren't restarted as there is no package avaialable for installation

# Knife

- Executes queries which resolve to resources and initiates a chef-client run
- *O Mighty Cloud, give unto me the set of machines that are part of the facebook analytics staging group and trigger chef-client*

```
knife
ssh -a cloud.public_hostname
"chef_environment:staging AND role:facebook_analytics_service
"sudo chef-client_withlock"
```

# chef-client

- Fetches the latest from the chef repo, runs chef client and executes definitions defined in the role/recipe

# Development

Testing cookbooks with vagrant

1. Create a Vagrantfile (ex. in confluence)
2. Specify the role:

```
chef.add_role("facebook_analytics_service_dev")
```

3. `$ vagrant up`
4. `$ vagrant halt`
5. Update your role/cookbook/attributes etc
6. `$ vagrant provision` or `$ vagrant destroy -f; vagrant up`

# Development

No bueno? ssh in and debug

```
$ vagrant ssh
```

# Development

When you're ready to submit,

- Open up a PR with your changes
- Remember to bump the version in METADATA.rb of any related cookbooks

# Other

- Inheritence
  - socialcode_service
- Chef overview
- Questions?