Introduction to Chef Automation

Introduce Yourselves

- Name
- Current job role
- Previous job roles/background
- Experience with Chef and/or config management
- Favorite Text Editor

Expectations

You will leave this class with a basic understanding of Chef's core components, architecture, commonly used tools, and basic troubleshooting methods

You bring with you your own domain expertise and problems. Chef is a framework for solving those problems. Our job is to teach you how to express solutions to your problems with Chef.

Course Objectives

After completing this course, you should be able to:

- Use Chef Resources to define the state of your system
- Use the Chef Development Kit to build and test cookbooks
- Understand the concepts of building reusable cookbooks across your organization

Agenda

Getting a Workstation
Resources and recipes
Writing a web server cookbook
Testing with Test Kitchen
Details About a System
Desired State and Data

Chef

Chef can automate how you build, deploy, and manage your infrastructure.

Chef can integrate with cloud-based platforms such as Azure and Amazon Elastic Compute Cloud to automatically provision and configure new machines.

Chef

Chef is a large set of tools that are able to be used on multiple platforms and in numerous configurations.

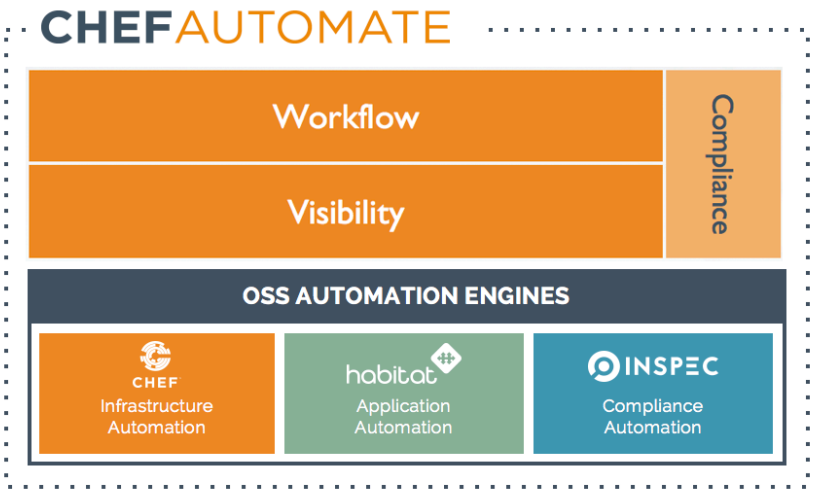
Learning Chef is like learning a language. You will learn the basic concepts very fast but it will take practice until you become comfortable.

A great way to learn Chef is to use Chef

CHEF

CHEF

High Level Chef Architecture



Chef is...

- An automation framework that enables Infrastructure as Code
- A robust set of tooling for testing Chef code
- A large library of reusable patterns (supermarket.chef.io)
- Available for Linux variants, Unix variants, and Windows, all as first class citizens.

CHEF IS INFRASTRUCTURE AS CODE

Programmatically provision and configure components	Treat like any other code base	Reconstruct business from code repository, data backup, and compute resources

The Chef DSL (domain specific language)

- Programmatically provision and configure components
- Declarative DSL with the flexibility
- Built on Ruby
- Extensible through Ruby

Treated Like Any Other Codebase…

- Stored in source control
- Testing Coverage
- Part of your CI or CD pipelines

Core Chef Concepts

Building Blocks: What is a Resource?

- A Resource is a system state you define
  - Example: Package installed, state of a service, configuration file existing
- You declare what the state of the resource is
  - Chef automatically determine HOW that state is achieved

```
windows_feature 'IIS-WebServerRole' do
          action :install
end
```

```
package 'httpd' do
          action :install
end
```

Building Blocks: What is a Recipe?

- A recipe is a collection of Resources
- Resources are executed in the order they are listed

```
windows_feature 'IIS-WebServerRole' do
  action :install
end

template 'c:\inetpub\wwwroot\Default.htm' do
  source 'Default.htm.erb'
  rights :read, 'Everyone'
end

service 'w3svc' do
  action [ :enable, :start ]
end
```

```
package 'httpd' do
  action :install
end

template '/var/www/index.html' do
  source 'index.html.erb'
  mode '0644'
end

service 'httpd' do
  action [ :enable, :start ]
end
```

Building Blocks: What is a Cookbook?

- A cookbook is a set of recipes
- A cookbook is a defined set of items and different outcomes that you expect to address
  - A cookbook could have a recipe to install apache2/httpd but also another set of recipes to activate modules required.

```
./attributes
./attributes/default.rb
./CHANGELOG.md
./metadata.rb
./README.md
./recipes
./recipes/application.rb
./recipes/balancer.rb
./recipes/database.rb
./recipes/default.rb
./recipes/webserver.rb
./templates
./templates/default
./templates/default/mysite.conf.erb
```

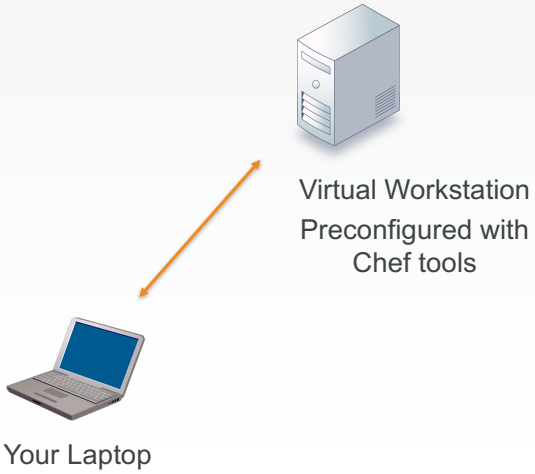
Chef Fundamentals

Ask Me Anything: It is important that we answer your questions and set you on the path to find more.

Break It: If everything works the first time go back and make some changes. Break it!

Chef Lab System Architecture

Lab Architecture



Hands-on Legend

- GL or Group Lab: All participants and the instructor do this task together with the instructor often leading the way and explaining things as we proceed.
- Lab: You perform this task on your own.

LAB

Group Lab:
Pre-built Workstation

We will provide for you a workstation with all the tools installed.

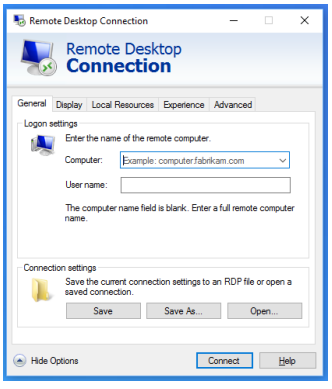
OBJECTIVE:

- ❑ Login to the Remote Workstation

CHEF

GL: Login to the Remote Workstation

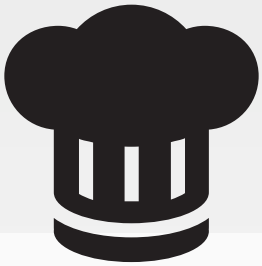
Use the **address**, **user name**, and **password** to connect to the remote workstation.



CHEF

LAB

Group Lab:
Pre-built Workstation



We will provide for you a workstation with all the tools installed.

OBJECTIVE:

- ✓ Login to the Remote Workstation

CHEF

Getting a Workstation

The chef user has been granted password-less sudoers access

The following software is installed on the remote workstation:

- Chef DK
- Atom, Visual Studio Code
- kitchen-ec2 gem
- Inspec gem

CHEF

Chef Resources

Chef's Fundamental Building Blocks

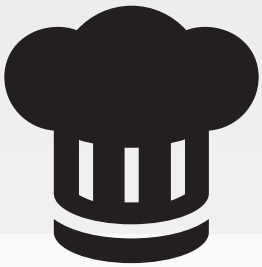
Objectives

After completing this module, you should be able to:

- Define Chef Resources
- Use the chef-client command
- Create a basic Chef recipe file

LAB

Group Lab: Hello, World?



I heard Chef is written in Ruby. If that's the case it's required that we write a quick "Hello, world!" application.

OBJECTIVE:

- ❑ Create a recipe file writes out 'Hello, world!' to a text file
- ❑ Apply the recipe to the workstation

Learning Chef

One of the best ways to learn a technology is to apply the technology in every situation that it can be applied.

A number of chef tools are installed on the system so lets put them to use.

Choose an Editor

You'll need to choose an editor to edit files:

Tips for using these editors can be found below in your participant guide.

atom

Visual Studio Code (← this one is awesome)

DOCS

Resources

A resource is a statement of configuration policy.

It describes the desired state of an element of your infrastructure and the steps needed to bring that item to the desired state.

https://docs.chef.io/resources.html

CHEF

Example: Package

```
package 'httpd' do
  action :install
end
```

The package named 'httpd' is installed.

https://docs.chef.io/resource_package.html

CHEF

Example: Service

```
service 'ntp' do
  action [ :enable, :start ]
end
```

The service named 'ntp' is enabled (start on reboot) and started.

https://docs.chef.io/resource_service.html

Example: File

```
file '/etc/motd' do
  content 'This computer is the property...'
end
```

The file name '/etc/motd' is created with content 'This computer is the property ...'

https://docs.chef.io/resource_file.html

Example: File

```
file '/etc/php.ini.default' do
  action :delete
end
```

The file name '/etc/php.ini.default' is deleted.

https://docs.chef.io/resource_file.html

CHEF

Working within Home Directory

```
> cd ~
```

CHEF

GL: Create and Open a Recipe File

```
> code hello.rb
```

 35

GL: Create a Recipe File Named hello.rb

~\hello.rb

```
file 'C:\hello.txt' do
  content 'Hello, world!'
end
```

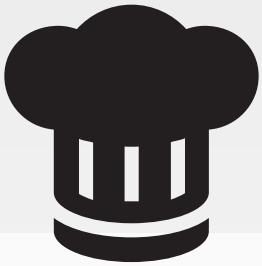
The file named ’c:\hello.txt' is created with the content 'Hello, world!'

https://docs.chef.io/resources.html

 2-36

LAB

Group Lab: Hello, World?



I heard Chef is written in Ruby. If that's the case it's required that we write a quick "Hello, world!" application.

OBJECTIVE:

- ✓ Create a recipe file writes out 'Hello, world!' to a text file
- ❑ Apply the recipe to the workstation

CHEF

CONCEPT

chef-client

chef-client is an agent that runs locally on every node that is under management by Chef.

When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state.

https://docs.chef.io/chef_client.html

CHEF

CONCEPT

chef-apply

chef-apply is an executable program that runs a single recipe from the command line

CHEF

GL: Apply a Recipe File

```
> chef-apply hello.rb -l info
```

```
[2017-08-11T04:00:38+00:00] INFO: Run List is []
[2017-08-11T04:00:38+00:00] INFO: Run List expands to []
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * file[C:\hello.txt] action create[2017-08-11T04:00:38+00:00] INFO: Processing
file[C:\hello.txt] action create ((chef-apply cookbook)::(chef-apply recipe) line
1)
[2017-08-11T04:00:38+00:00] INFO: file[C:\hello.txt] created file C:\hello.txt

    - create new file C:\hello.txt[2017-08-11T04:00:38+00:00] INFO:
file[C:\hello.txt] updated file contents C:\hello.txt


    - update content in file C:\hello.txt from none to 315f5b
    --- C:\hello.txt    2017-08-11 04:00:38.000000000 +0000
```


CHEF

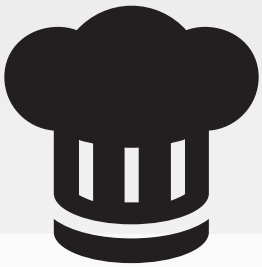
GL: What Does hello.txt Say?

```
> gc C:\hello.txt
```

```
Hello, world!
```


LAB

Group Lab: Hello, World?



I heard Chef is written in Ruby. If that's the case it's required that we write a quick "Hello, world!" application.

OBJECTIVE:

- ✓ Create a recipe file writes out 'Hello, world!' to a text file
- ✓ Apply the recipe to the workstation

Discussion

1. What would happen if you applied the recipe again?

2. What would happen if the package were to become uninstalled?

EXERCISE

Lab: Test and Repair

What would happen if the file contents were modified?

- ❑ Modify the contents of 'C:\hello.txt' manually with your text editor
- ❑ Run the chef-client command again

Modify the Contents of the File

C:\hello.txt

```
Goodbye, world!
```


CHEF

Re-apply the Policy Defined in the Recipe

```
> chef-apply hello.rb -l info
```

```
[2017-08-11T04:02:45+00:00] INFO: file[C:\hello.txt] updated file contents
C:\hello.txt

    - update content in file C:\hello.txt from f701ac to 315f5b
    --- C:\hello.txt    2017-08-11 04:02:33.000000000 +0000
    +++ C:\chef-hello20170811-2148-13txos5.txt  2017-08-11 04:02:45.000000000
+0000
    @@ -1,2 +1,2 @@
    -Adios world!
    +Hello, world!
```


CHEF

EXERCISE

Lab: Test and Repair

What would happen if the file contents were modified?

- ✓ Modify the contents of 'C:\hello.txt' manually with your text editor
- ✓ Run the chef-client command again

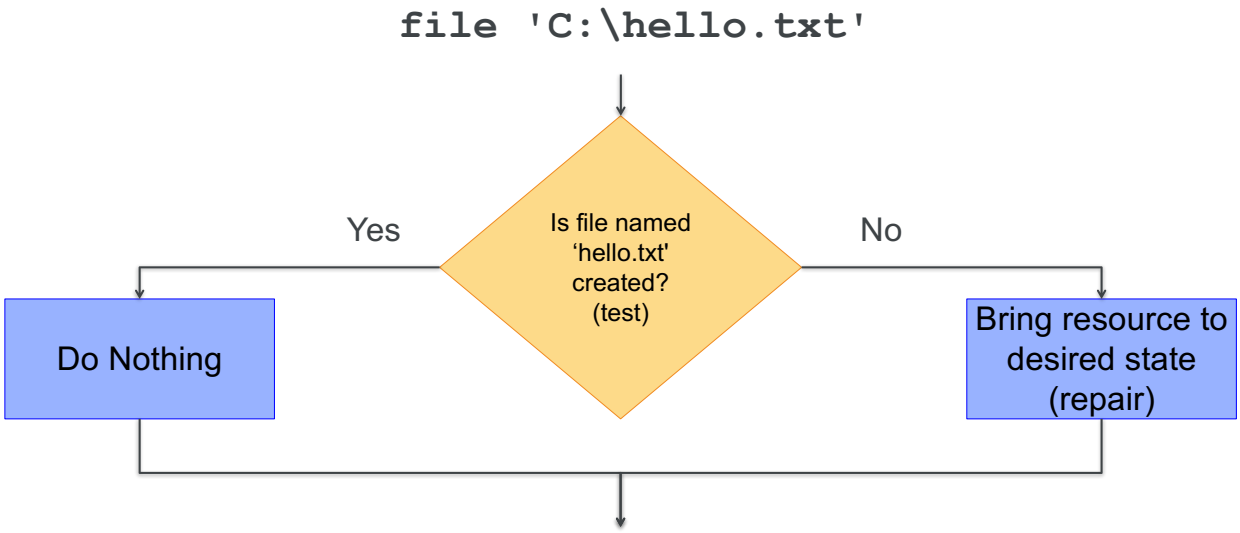
CONCEPT

Test and Repair

`chef-client` takes action only when it needs to. Think of it as test and repair.

Chef looks at the current state of each resource and takes action only when that resource is out of policy.

Test and Repair



Test and Repair

What would happen if the file permissions (mode), owner, or group changed?

Have we defined a policy for these properties?

CONCEPT

Resource Definition

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with PROPERTIES

CHEF

CONCEPT

Resource Definition

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with PROPERTIES

CHEF

CONCEPT

Resource Definition

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with PROPERTIES

CHEF

CONCEPT

Resource Definition

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with PROPERTIES

CHEF

CONCEPT

Resource Definition

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

?

The TYPE named NAME should be **ACTION'd** with PROPERTIES

CHEF

EXERCISE

Lab: The `file` Resource

- ❑ **Read** https://docs.chef.io/resources.html
- ❑ **Discover the file resource's:**
  - default action
  - **rights** attribute
- ❑ **Update the `file` policy in "hello.rb" to:**

  The file named 'C:\hello.txt' is created with 'read' rights for 'Everyone'.

CHEF

Lab: The Updated file Resource

~\hello.rb

```
file 'C:\hello.txt' do
  content 'Hello, world!'
  rights :read, 'Everyone'
  action :create
end
```

This sets the file's rights to allow 'Everyone' access.

The default action is to create (not necessary to define it).

EXERCISE

Lab: The `file` Resource

- ✓ **Read** https://docs.chef.io/resources.html
- ✓ **Discover the file resource's:**
  - default action
  - **rights** attribute
- ✓ **Update the `file` policy in "hello.rb" to:**

  The file named 'C:\hello.txt' is created with 'read' rights for 'Everyone'.

Questions

What questions can we answer for you?

 59

Discussion

What is a resource?

What are some other possible examples of resources?

How did the example resources we wrote describe the desired state of an element of our infrastructure?

What does it mean for a resource to be a statement of configuration policy?

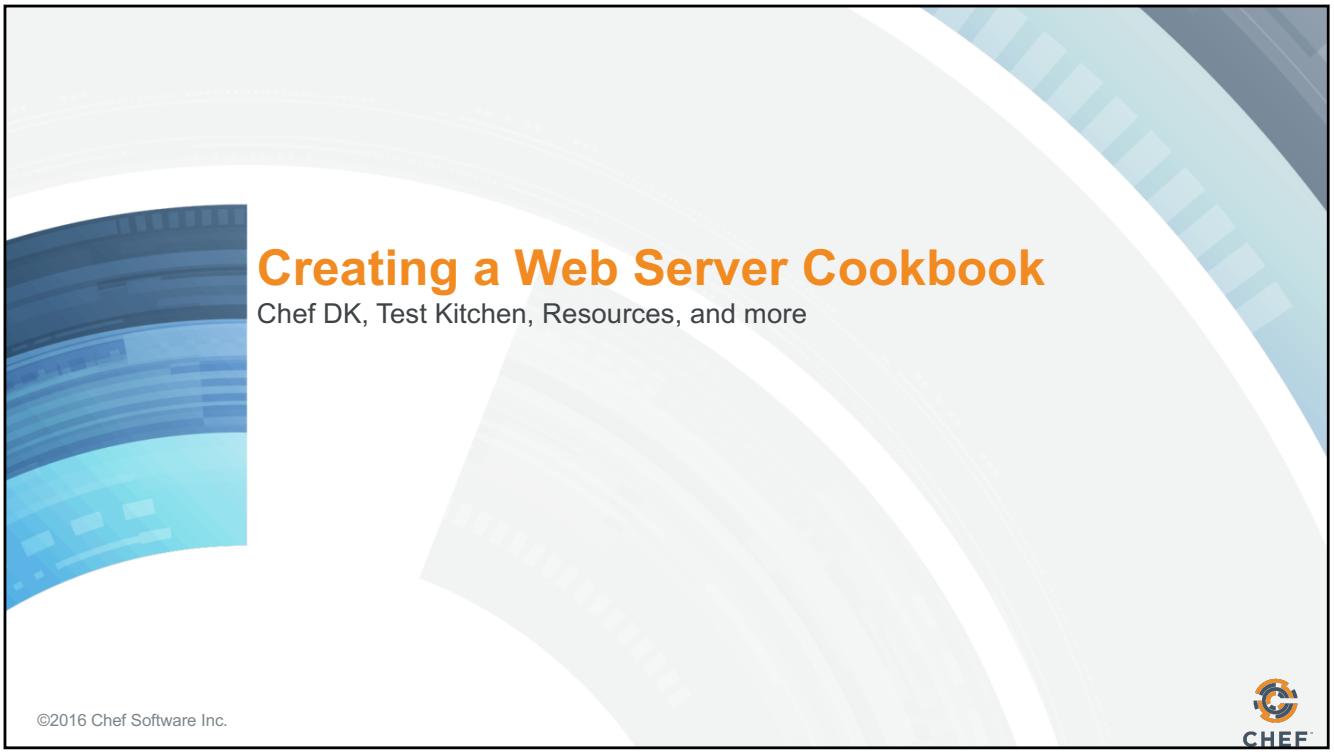
 60

Q&A

What questions can we answer for you?

- chef-client
- chef-apply
- Resources
- Resource - default actions and default properties
- Test and Repair

CHEF™



Objectives

After completing this module, you should be able to:

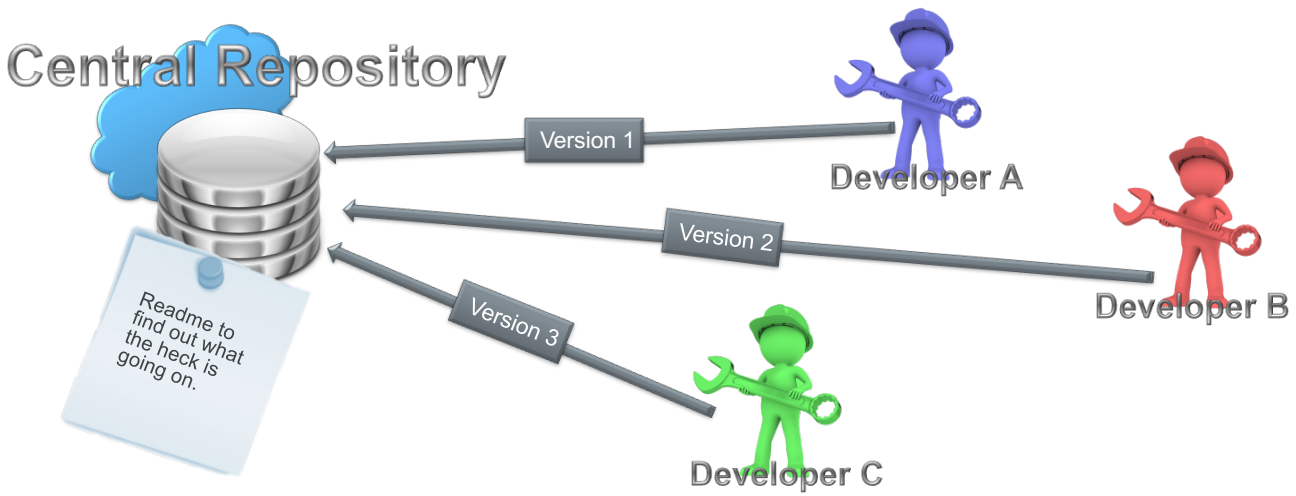
- Generate a Chef cookbook
- Define a Chef recipe that sets up a web server
- Use Test Kitchen to verify your recipes converge on a virtual instance
- Use Inspec to test your cookbook

CHEF

Questions You May Have

1. Thinking about recipes, could we do something like that for a web server?
2. Is there a way to package up recipes you create with a version number (and maybe a README)?
3. I think `chef` is able to generate something called a cookbook. Shouldn't we start thinking about some version control so we don't lose all our hard work?

Collaboration and Version Control



Versioning Pros and Cons

```
$ cp setup.rb setup.rb.bak
or
$ cp setup{,.`date +%Y%m%d%H%M`}
or
$ cp setup{,.`date +%Y%m%d%H%M`-`$USER`}
```

Saving a copy of the original file as another filename.

CHEF

Git Version Control

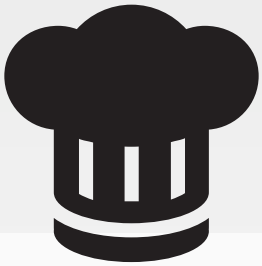
git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.

We will be using **git** throughout the rest of this course.

CHEF

LAB

GL: Create a Cookbook



How are we going to manage this file? Does it need a README?

OBJECTIVE:

- ❑ Use chef to generate a cookbook
- ❑ Add the new cookbook to version control

CHEF

Cookbooks

A Chef cookbook is the fundamental unit of configuration and policy distribution.

Each cookbook defines a scenario, such as everything needed to install and configure MySQL, and then it contains all of the components that are required to support that scenario.

Read the first three paragraphs here: http://docs.chef.io/cookbooks.html

CHEF

Working within Home Directory

```
> cd ~
```

GL: Create a Cookbooks Directory

```
> mkdir cookbooks
```

CONCEPT

What is 'chef'?

An executable program that allows you generate cookbooks and cookbook components.

CHEF

What can 'chef' do?

```
> chef --help
```

```
UsaGL:
    chef -h/--help
    chef -v/--version
    chef command [arguments...] [options...]

Available Commands:
    exec        Runs the command in context of the embedded ruby
    gem         Runs the `gem` command in context of the embedded ruby
    generate    Generate a new app, cookbook, or component
    shell-init  Initialize your shell to use ChefDK as your primary ruby
    install     Install cookbooks from a Policyfile and generate a locked cookboo...
    update      Updates a Policyfile.lock.json with latest run_list and cookbooks
```


CHEF

What Can 'chef generate' Do?

```
> chef generate --help
UsaGL: chef generate GENERATOR [options]

Available generators:
  app          Generate an application repo
  cookbook     Generate a single cookbook
  recipe       Generate a new recipe
  attribute    Generate an attributes file
  template     Generate a file template
  file         Generate a cookbook file
  lwrp         Generate a lightweight resource/provider
  repo         Generate a Chef policy repository
  policyfile   Generate a Policyfile for use with the install/push commands
  generator    Copy ChefDK's generator cookbook so you can customize it
```


GL: Let's Create a Cookbook

```
> chef generate cookbook cookbooks/webserver
Generating cookbook webserver
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master

Your cookbook is ready. Type `cd cookbooks/webserver` to enter it.

There are several commands you can run to get started locally developing and
testing your cookbook.
Type `delivery local --help` to see a full list.
```


GL: The Cookbook Has a README

```
> tree /f cookbooks\webserver
webserver
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│ ...
6 directories, 8 files
```


CONCEPT

README.md

The description of the cookbook's features written in Markdown.

http://daringfireball.net/projects/markdown/syntax

GL: The Cookbook Has Some Metadata

```
> tree /f cookbooks\webserver
webserver
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
| ...
6 directories, 8 files
```


DOCS

metadata.rb

Every cookbook requires a small amount of metadata. Metadata is stored in a file called metadata.rb that lives at the top of each cookbook's directory.

http://docs.chef.io/config_rb_metadata.html

GL: Let's Take a Look at the Metadata

```
> gc cookbooks\webserver\metadata.rb
```

```
name             'webserver'
maintainer       'The Authors'
maintainer_email 'you@example.com'
license          'all_rights'
description      'Installs/Configures webserver'
long_description 'Installs/Configures webserver'
version          '0.1.0'

# If you upload to Supermarket you should set this so your cookbook
# gets a `View Issues` link
# issues_url 'https://github.com/<insert_org_here>/webserver/issues' if
respond_to?(:issues_url)
```


GL: The Cookbook Has a Folder for Recipes

```
> tree /f cookbooks\webserver
```

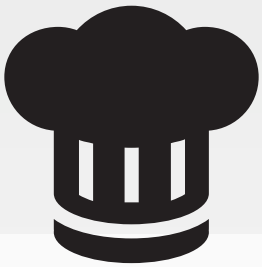
```
webserver
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
| ...
6 directories, 8 files
```


GL: The Cookbook Has a Default Recipe

```
> gc cookbooks\webserver\recipes\default.rb
#
# Cookbook:: webserver
# Recipe:: default
#
# Copyright:: 2017, The Authors, All Rights Reserved.
```


LAB

GL: Create a Cookbook



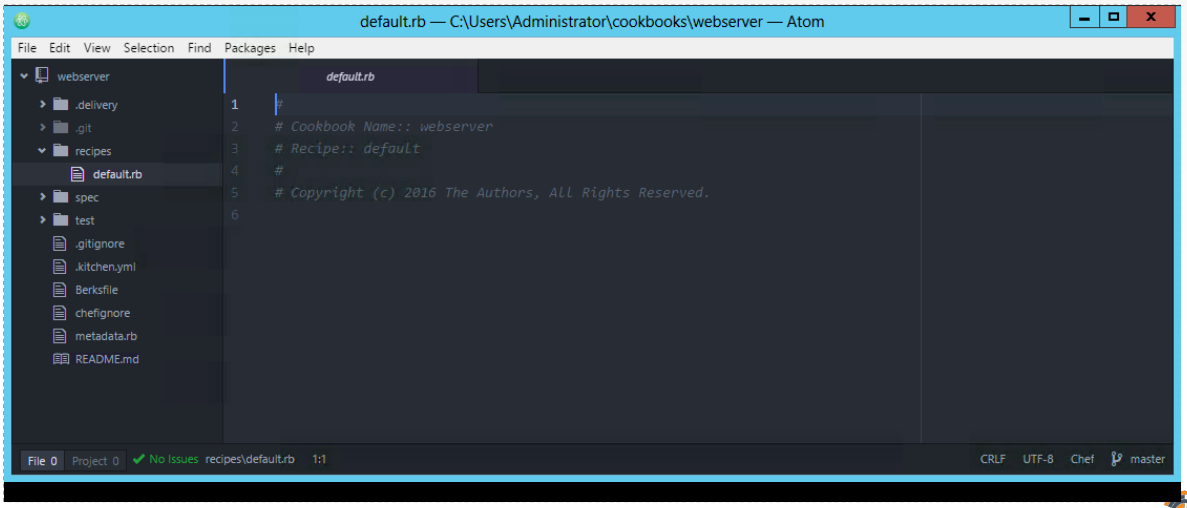
How are we going to manage this file? Does it need a README?

OBJECTIVE:

- ✓ Use chef to generate a cookbook
- ❑ Add the new cookbook to version control

GL: Use Your Editor to Open the Cookbook

```
> code cookbooks\webserver
```



 85

GL: Update the Default Recipe

cookbooks\webserver\recipes\default.rb

```
package 'httpd' do
  action :install
end

file '/var/www/html/index.html' do
  content '<h1>Hello, world!</h1>'
end

service 'httpd' do
  action [:enable, :start]
end
```

 2-86

GL: Move into the Cookbook Directory

```
> cd cookbooks\webserver
```


GL: Initialize the Directory as a git Repository

```
> git init
Reinitialized existing Git repository in /home/chef/cookbooks/webserver/.git/
```


GL: Use 'git add' to Stage Files to be Committed

```
> git add .
```


CHEF

CONCEPT

Staging Area

The staging area has a file, generally contained in your Git directory, that stores information about what will go into your next commit.

It's sometimes referred to as the "index", but it's also common to refer to it as the staging area.

http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

CHEF

GL: Use 'git status' to View the Staged Files

```
> git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   .kitchen.yml
        new file:   Berksfile
        new file:   README.md
        new file:   chefignore
        new file:   metadata.rb
```

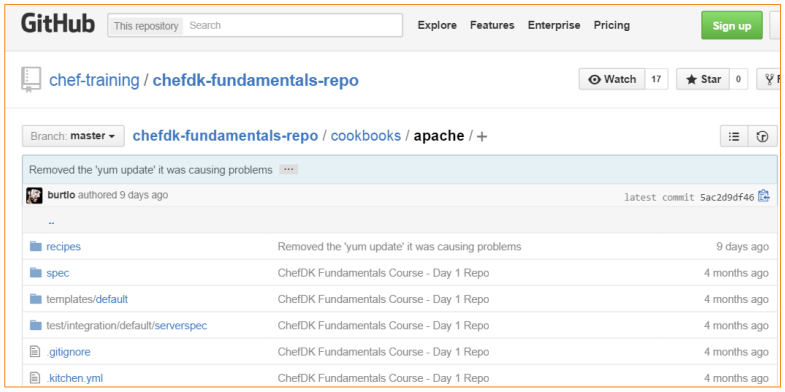

GL: Use 'git commit' to Save the Staged Changes

```
> git commit -m "Initial commit"
master (root-commit) 9998472] Initial webserver cookbook
 Committer: ChefDK User <chef@ip-172-31-59-191.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author
```


Git Version Control

If you use git versioning you should ultimately push the local git repository to a shared remote git repository.

In this way others could collaborate with you from a centralized location.



GL: Return to the Home Directory

```
> cd ~
```

Can We Test Cookbooks?

As we start to define our infrastructure as code we also need to start thinking about testing it.

CHEF

Mandating Testing

What steps would it take to test one of the cookbooks that we have created?

CHEF

Steps to Verify Cookbooks

Testing Cookbooks

We can start by first mandating that all cookbooks are tested

How often should you test your cookbook?

How often do you think changes will occur?

What happens when the rate of cookbook changes exceed the time interval it takes to verify the cookbook?

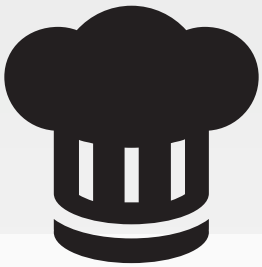
CONCEPT

Code Testing

An automated way to ensure code accomplishes the intended goal and help the team understand its intent

LAB

Test Configuration

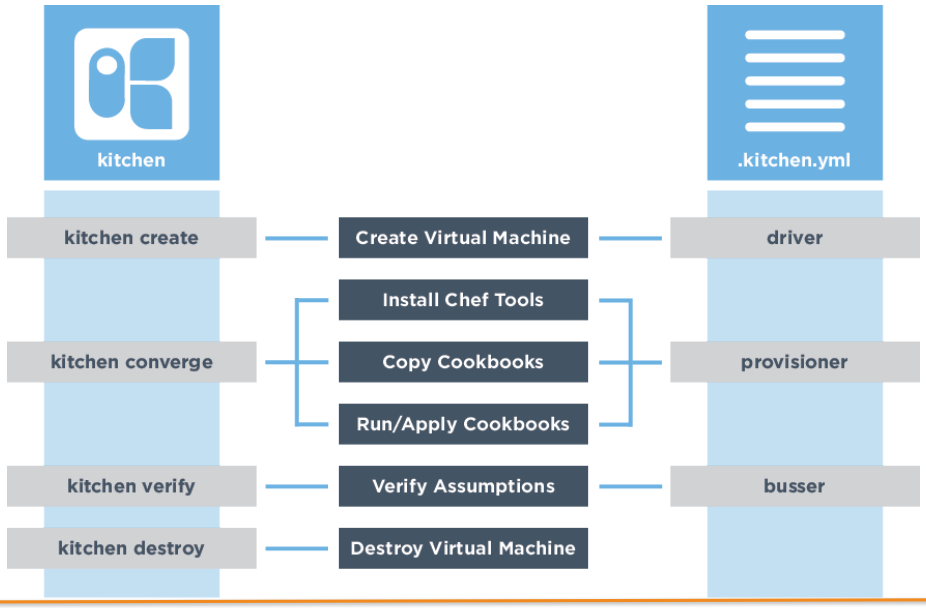


What are we running in production? Maybe I could test the cookbook against a virtual machine.

OBJECTIVE:

- ❑ Configure the ”webserver" cookbook to test against the centos-6.7 platform
- ❑ Test the "webserver" cookbook on a virtual machine

Test Kitchen Commands and Configuration



What Can 'kitchen' Do?

```
> kitchen --help
```

```
Commands:
  kitchen console                          # Kitchen Console!
  kitchen converge [INSTANCE|REGEXP|all]   # Converge one or more instances
  kitchen create [INSTANCE|REGEXP|all]     # Create one or more instances
  kitchen destroy [INSTANCE|REGEXP|all]    # Destroy one or more instances
  ...
  kitchen help [COMMAND]                   # Describe available commands or one specif...
  kitchen init                             # Adds some configuration to your cookbook...
  kitchen list [INSTANCE|REGEXP|all]       # Lists one or more instances
  kitchen setup [INSTANCE|REGEXP|all]      # Setup one or more instances
  kitchen test [INSTANCE|REGEXP|all]       # Test one or more instances
  kitchen verify [INSTANCE|REGEXP|all]     # Verify one or more instances
  kitchen version                          # Print Kitchen's version information
```

Do We Have a .kitchen.yml?

```
> tree /f cookbooks\webserver
```

```
...
├── .kitchen.yml
├── metadata.rb
├── README.md
├── recipes
│   ├── default.rb
│   └── setup.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
└── test
```


What is Inside .kitchen.yml?

```
> gc cookbooks\webserver\.kitchen.yml
```

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```


CONCEPT

.kitchen.yml

When chef generates a cookbook, a default .kitchen.yml is created. It contains kitchen configuration for the driver, provisioner, platform, and suites.

http://kitchen.ci/docs/getting-started/creating-cookbook

CHEF

Demo: The kitchen Driver

~/cookbooks/webserver/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

The driver is responsible for creating a machine that we'll use to test our cookbook.

Example Drivers:

- docker
- vagrant

CHEF

Demo: The kitchen Provisioner

~\cookbooks\webserver\.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

This tells Test Kitchen how to run Chef, to apply the code in our cookbook to the machine under test.

The default and simplest approach is to use chef_zero.

CHEF

Demo: The kitchen Verifier

~\cookbooks\webserver\.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

This tells Test Kitchen how to verify the converged instances.

The default approach is to use InSpec.

CHEF

Demo: The kitchen Platforms

~\cookbooks\webserver\.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

This is a list of operation systems on which we want to run our code.

Demo: The kitchen Suites

~\cookbooks\webserver\.kitchen.yml

```
...

suites:
  - name: default
    run_list:
      - recipe[webserver::default]
    verifier:
      inspec_tests:
        - test/recipes
    attributes:
```

This section defines what we want to test. It includes the Chef run-list of recipes that we want to test.

We define a single suite named "default".

Demo: The kitchen Suites

~\cookbooks\webserver\.kitchen.yml

```
...

suites:
  - name: default
    run_list:
      - recipe[webserver::default]
    verifier:
      inspec_tests:
        - test/smoke/default
    attributes:
```

The suite named "default" defines a run_list.

Run the ”webserver" cookbook's "default" recipe file.

CONCEPT

Kitchen Test Matrix

Kitchen defines a list of instances, or test matrix, based on the platforms multiplied by the suites.

PLATFORMS x SUITES

Running kitchen list will show that matrix.

Example: Kitchen Test Matrix

```
> kitchen list

Instance             Driver   Provisioner  Verifier  Transport Last Action
default-ubuntu-1204  Vagrant  ChefZero     Busser    Ssh       <Not Created>
default-centos-65    Vagrant  ChefZero     Busser    Ssh       <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[webserver::default]
    attributes:
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```


Example: Kitchen Test Matrix

```
> kitchen list

Instance             Driver   Provisioner  Verifier  Transport Last Action
default-ubuntu-1204  Vagrant  ChefZero     Busser    Ssh       <Not Created>
default-centos-65    Vagrant  ChefZero     Busser    Ssh       <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[webserver::default]
    attributes:
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```


LAB

Group Exercise: Test Configuration

What are we running in production? Maybe I could test the cookbook against a virtual machine.

OBJECTIVE:

- Configure the "webserver" cookbook's .kitchen.yml to use the EC2 driver and centos 6.x platform
- Use kitchen converge to apply the recipe on a virtual machine

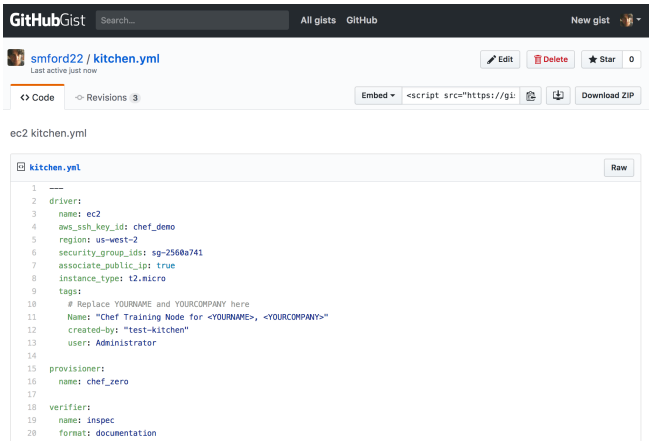
CHEF

GL: Move into the Cookbook's Directory

```
> cd cookbooks\webserver
```


CHEF

Copy/Replace .kitchen.yml



https://goo.gl/Nv882w

CHEF

GL: Copy .kitchen_linux.yml template

```
> cp ~/Desktop/Test_Kitchen/kitchen_linux.yml ~/cookbooks/webserver/.kitchen.yml
```


CHEF

GL: Edit the Kitchen Configuration File

~/cookbooks/webserver/.kitchen.yml

```
...
platforms:
  - name: centos-6
    transport:
      username: root
      ssh_key: C:\Users\Administrator\.ssh\id_rsa
    driver_config:
      user_data: C:\Users\Administrator\user_data
  - name: centos-7
    transport:
      username: root
      ssh_key: C:\Users\Administrator\.ssh\id_rsa
...
```

https://www.centos.org

CHEF

GL: Edit the Kitchen Configuration File

~/cookbooks/webserver/.kitchen.yml

```
...
suites:
  - name: default
    run_list:
      - recipe[webserver::default]
    verifier:
      inspec_tests:
        - test/smoke/default
    attributes:
```


CHEF

GL: Edit the Kitchen Configuration File

~/cookbooks/webserver/.kitchen.yml

```
...
platforms:
  - name: centos-6
    transport:
      username: root
      ssh_key: C:\Users\Administrator\.ssh\id_rsa
    driver_config:
      user_data: C:\Users\Administrator\user_data
  - name: centos-7
    transport:
      username: root
      ssh_key: C:\Users\Administrator\.ssh\id_rsa
...
```

https://www.centos.org

CHEF

GL: Edit the Kitchen Configuration File

~/cookbooks/webserver/.kitchen.yml

```
...
suites:
  - name: default
    run_list:
      - recipe[webserver::default]
    verifier:
      inspec_tests:
        - test/recipes
    attributes:
```


CHEF

GL: Look at the Test Matrix

```
> kitchen list
```

Instance	Driver	Provisioner	Verifier	Transport	Last Action
default-centos-6	Ec2	ChefZero	Inspec	Ssh	<Not Created>

LAB

Converging a Cookbook

Before I add features it really would be nice to test these cookbooks against the environments that resemble production.

OBJECTIVE:

- ✓ Configure the "webserver" cookbook's .kitchen.yml to use the ec2 driver and centos-6.x platform
- ❑ Use kitchen converge to apply the recipe on a virtual machine

Kitchen Create


```
> kitchen create [INSTANCE|REGEXP|all]

Create one or more instances.
```


Group Exercise: Kitchen Converge


```
> kitchen converge [INSTANCE|REGEXP|all]

Create the instance (if necessary) and then apply
the run list to one or more instances.
```


GL: Converge the Cookbook

```
> kitchen converge
```

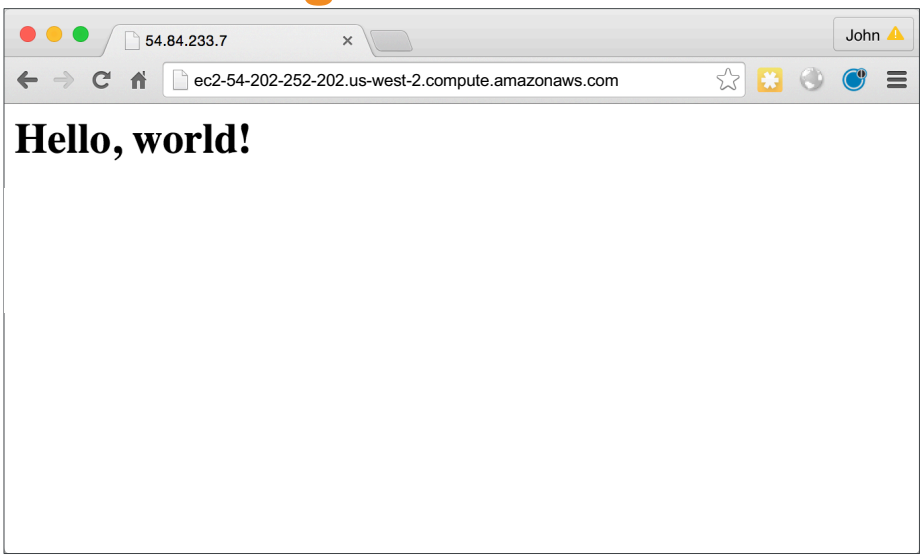
```
-----> Starting Kitchen (v1.13.2)
-----> Creating <default-centos-6>...
       Detected platform: centos version 6 on x86_64. Instance Type: t2.micro.
Default username: centos (default).
       If you are not using an account that qualifies under the AWS
free-tier, you may be charged to run these suites. The charge
should be minimal, but neither Test Kitchen nor its maintainers
are responsible for your incurred costs.

       Instance <i-b2bb691c> requested.
       Polling AWS for existence, attempt 0...
       Attempting to tag the instance, 0 retries
       EC2 instance <i-b2bb691c> created.
```


GL: Retrieve the FQDN for your instance

```
> gc .kitchen\default-centos-6.yml
```

```
---
username: centos
server_id: i-d63ba942
hostname: ec2-54-202-252-202.us-west-2.compute.amazonaws.com
last_action: converge
```


GL: Testing Our Websites



Test Kitchen

What is being tested when kitchen converges a recipe without error?

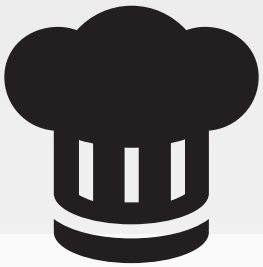
What is NOT being tested when kitchen converges the recipe without error?

Test Kitchen

What is left to validate to ensure that the cookbook successfully applied the policy defined in the recipe?

LAB

The First Test



Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?

OBJECTIVE:

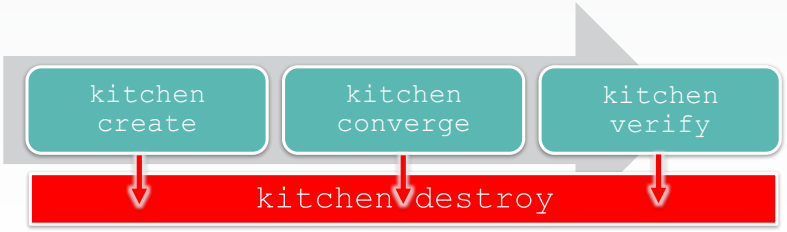
- ❑ In a few minutes we'll write and execute a test that asserts that the httpd package is installed when the "webserver" cookbook's default recipe is applied.

Kitchen Verify


```
> kitchen verify [INSTANCE|REGEXP|all]
```

Create, converge, and verify one or more instances.

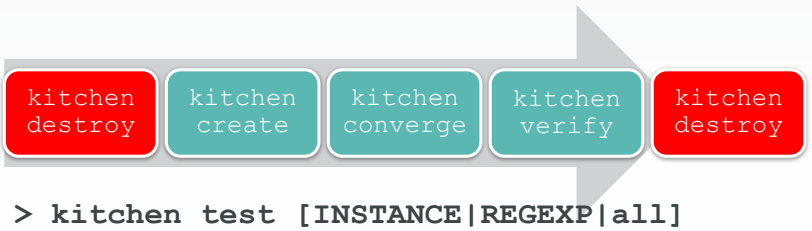
Kitchen Destroy




```
> kitchen destroy [INSTANCE|REGEXP|all]
```

Destroys one or more instances.

Kitchen Test




```
> kitchen test [INSTANCE|REGEXP|all]

Destroys (for clean-up), creates, converges, verifies
and then destroys one or more instances.
```


InSpec

InSpec tests your servers' actual state by executing command locally, via SSH, via WinRM, via Docker API and so on.

https://inpsec.io

Example: Is the 'tree' Package Installed?

```
describe package('tree') do
  it { should be_installed }
end
```

I expect the package tree should be installed.

https://docs.chef.io/inspec_reference.html#id118

Example: Is the Port 80?

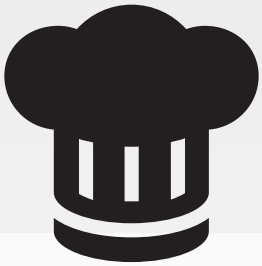
```
describe port(80) do
  it { should be_listening }
end
```

I expect port 80 to be listening.

https://docs.chef.io/inspec_reference.html#id118

LAB

Testing Our Webserver



I would love to know that the webserver is installed and running correctly.

OBJECTIVE:

- ❑ Discuss and decide what should be tested with the webserver cookbook

 139

Testing

What are some things we could test to validate our web server has deployed correctly?

What manual tests do we use now to validate a working web server?

 140

EXERCISE

Lab: Testing Webserver

- Update the test file for the ”webserver" cookbook's default recipe
- Add tests that validate a working web server
  https://www.inspec.io/docs/reference/resources/port

  https://www.inspec.io/docs/reference/resources/command
- Run kitchen verify
- Commit your changes

 141

Lab: Return Home and 'cd cookbooks/webserver'

```
> cd ~\cookbooks\webserver
```

 142

Lab: What Does the Webserver Say?

~\cookbooks\webserver\test\smoke\default\default_test.rb

```
unless os.windows?
  describe user('root') do
    it { should exist }, :skip do
  end
end
```



```

describe port(80) do
  it { should be_listening }, :skip do
end
```


Lab: What Does the Webserver Say?

~\cookbooks\webserver\test\smoke\default\default_test.rb

```

describe port(80) do
  it { should be_listening }
end
```



```
describe command('curl localhost') do
  its('stdout') { should match('Hello, world') }
end
```


GL: Use 'git add' to Stage Files to be Committed

```
> git add .
```


GL: Use 'git status' to View the Staged Files

```
> git status
warning: LF will be replaced by CRLF in .kitchen.yml.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in .kitchen.yml.
The file will have its original line endings in your working directory.
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   .kitchen.yml
        modified:   test/recipes/default_test.rb
```


GL: Use 'git commit' to Save the Staged Changes

```
> git commit -m "Created initial tests"
master (root-commit) 9998472] Initial webserver cookbook
 Committer: ChefDK User <chef@ip-172-31-59-191.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author
```


Lab: Verifying the Expectations

```
> kitchen verify
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-67>...
       Use `/home/chef/cookbooks/webserver/test/recipes/default` for testing

Target:  ssh://kitchen@localhost:32769

  ✔  User root should exist
  ✔  Port 80 should be listening
  ✔  Command curl localhost stdout should match "Hello, world"

Summary: 3 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-67> (0m0.87s).
```


Discussion

Why do you have to run kitchen within the directory of the cookbook?

Where would you define additional platforms?

Why would you define a new test suite?

What are the limitations of using Test Kitchen to validate recipes?

Q&A

What questions can we help you answer?

- Test Kitchen
- kitchen commands
- kitchen configuration
- InSpec

Details About the System

Finding and Displaying Information About Our System

Objectives

After completing this module, you should be able to

- Capture details about a system
- Use the node object within a recipe
- Use Ruby's string interpolation
- Update the version of a cookbook

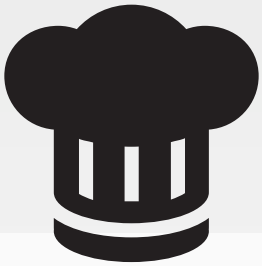
Managing a Large Number of Servers

Have you ever had to manage a large number of servers that were almost identical?

How about a large number of identical servers except that each one had to have host-specific information in a configuration file?

LAB

Details About the Node



Displaying system details in the home page definitely sounds useful.

OBJECTIVE:

- Update the index.html file contents, in the "webserver" cookbook, to include node details

CHEF

Some Useful System Data

- IP Address
- hostname
- memory
- CPU - MHz

CHEF

GL: Finding the IP Address

```
> ipconfig
```

```
Windows IP Configuration

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . : ec2.internal
   Link-local IPv6 Address . . . . . : fe80::2da8:4ba7:45e2:e863%21
   IPv4 Address. . . . . . . . . . . : 172.31.21.21
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . : 172.31.16.1
```


GL: Finding the Hostname

```
> hostname
```

```
WIN-KRQSVD3RFM7
```


GL: Finding the Total Memory

```
> wmic ComputerSystem get TotalPhysicalMemory
```

```
TotalPhysicalMemory
8052654080
```


GL: Finding the CPU MHz

```
> wmic cpu get name
```

```
Name
Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz
```


Example Cookbook updates

~\cookbooks\webserver\recipes\default.rb

```
file '/var/www/html/index.html' do
  content '<h1>Hello, world!</h1>

  <h2>IPADDRESS: 104.236.192.102</h2>
  <h2>HOSTNAME : banana-stand</h2>
  <h2>MEMORY   : 502272 kB</h2>
  <h2>CPU      : 2399.998 MHz</h2>
'
end
```


CHEF

Capturing System Data

What are the limitations of the way we captured this data?

How accurate will our data be when we deploy it on other systems?

Are these values we would want to capture in our tests?

CHEF

Hard Coded Values

The values that we have derived at this moment may not be the correct values when we deploy this recipe again even on the same system!

Data In Real Time

How could we capture this data in real-time?

CONCEPT

Ohai!

Ohai is a tool that already captures all the data that we similarly demonstrated finding.

http://docs.chef.io/ohai.html

CHEF

Ohai!

```
> ohai
```

```
{
  "kernel": {
    "name": "Linux",
    "release": "2.6.32-431.1.2.0.1.el6.x86_64",
    "version": "#1 SMP Fri Dec 13 13:06:13 UTC 2013",
    "machine": "x86_64",
    "os": "GNU/Linux",
    "modules": {
      "veth": {
        "size": "5040",
        "refcount": "0"
      },
      "ipt_addrtype": {
```


CHEF

CONCEPT

All About The System

Ohai queries the operating system with a number of commands, similar to the ones demonstrated.

The data is presented in JSON (JavaScript Object Notation).

http://docs.chef.io/ohai.html

CHEF

CONCEPT

ohai + chef-client = <3

chef-client and chef-apply automatically executes ohai and stores the data about the node in an object we can use within the recipes named node.

http://docs.chef.io/ohai.html

CHEF

CONCEPT

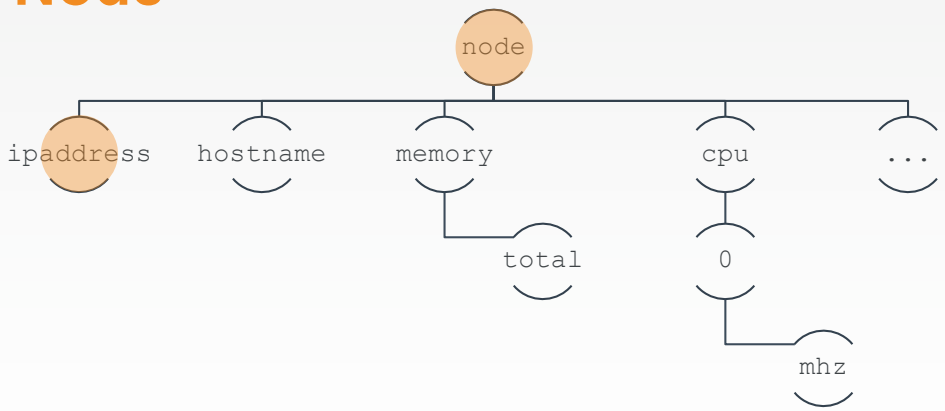
The Node Object

The node object is a representation of our system.
It stores all the attributes found about the system.

http://docs.chef.io/nodes.html#attributes

CHEF

The Node

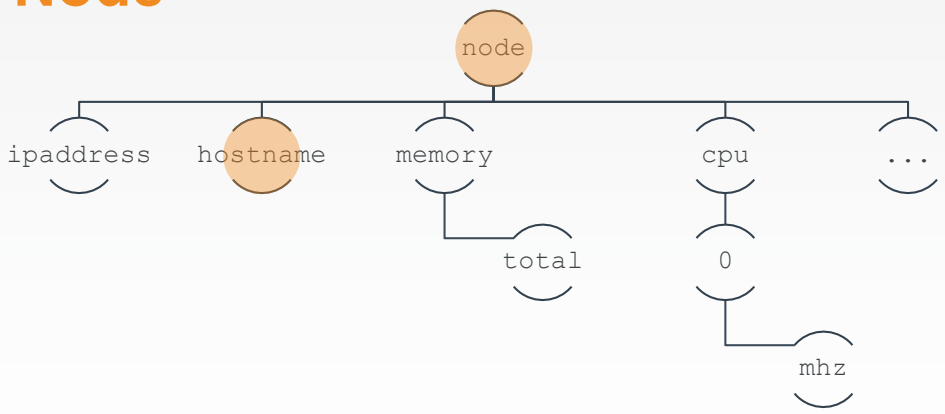



```
IPADDRESS: 104.236.192.102
```

```
"IPADDRESS: #{node['ipaddress']}"
```


CHEF

The Node

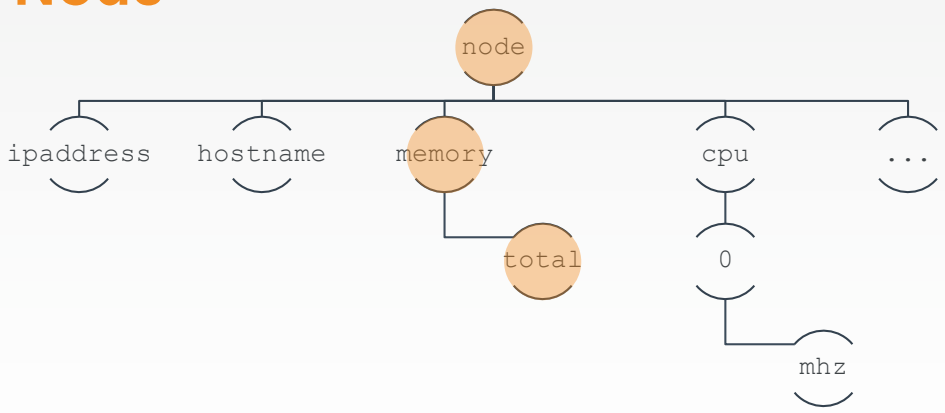



```
HOSTNAME: banana-stand
```

```
"HOSTNAME: #{node['hostname']}"
```


CHEF

The Node

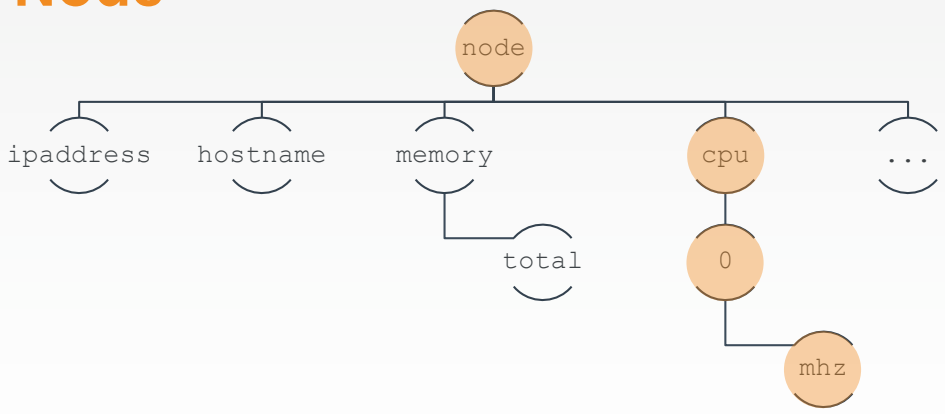



```
MEMORY: 502272kB
```

```
"Memory: #{node['memory']['total']}"
```


CHEF

The Node




```
CPU: 2399.998MHz
```

```
"CPU: #{node['cpu']['0']['mhz']}"
```


CONCEPT

String Interpolation

```
I have 4 apples
```

```
apple_count = 4
puts "I have #{apple_count} apples"
```

http://en.wikipedia.org/wiki/String_interpolation#Ruby

CONCEPT

String Interpolation

```
I have 4 apples
```

```
apple_count = 4
puts "I have #{apple_count} apples"
```

http://en.wikipedia.org/wiki/String_interpolation#Ruby

CHEF

CONCEPT

String Interpolation

I have 4 apples

apple_count = 4
puts "I have #{apple_count} apples"

CHEF

GL: Using the Node's Attributes

~\cookbooks\webserver\recipes\default.rb

```
...

file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>

  <h2>IPADDRESS: #{node['ipaddress']}</h2>
  <h2>HOSTNAME: #{node['hostname']}</h2>
  <h2>MEMORY: #{node['memory']['total']}</h2>
  <h2>CPU: #{node['cpu']['0']['mhz']}</h2>
"
end

...
```


CHEF

EXERCISE

GL: Verify the Changes

- Change directory into the ”webserver" cookbook's directory
- Run kitchen converge for the ”webserver" cookbook
- Run kitchen verify for the ”webserver" cookbook

CHEF

GL: Converge and Verify the Test Instance

```
> cd ~\cookbooks\webserver
> kitchen converge
> kitchen verify
```

```
...
-----> Starting Kitchen (v1.13.2)
-----> Converging <default-centos-6>...
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 5.1.0...
       Removing non-cookbook files before transfer
       Preparing validation.pem
       Preparing client.rb
-----> Chef Omnibus installation detected (install only if missing)
```


EXERCISE

GL: Verify the Changes

- ✓ Change directory into the ”webserver" cookbook's directory
- ✓ Run kitchen converge for the ”webserver" cookbook
- ✓ Run kitchen verify for the ”webserver" cookbook

LAB

Changes Mean a New Version

Let's bump the version number and check in the code to source control.

OBJECTIVE:

- ❑ Update the version of the ”webserver" cookbook
- ❑ Commit the changes to the ”webserver" cookbook to version control

CHEF

CONCEPT

Cookbook Versions

A cookbook version represents a set of functionality that is different from the cookbook on which it is based.

https://docs.chef.io/cookbook_versions.html

CHEF

CONCEPT

Semantic Versions

Given a version number MAJOR.MINOR.PATCH, increment the:

- MAJOR version when you make incompatible API changes
- MINOR version when you add functionality in a backwards-compatible manner
- PATCH version when you make backwards-compatible bug fixes

http://semver.org

CHEF

Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

CHEF

GL: Update the Cookbook Version

~\cookbooks\webserver\metadata.rb

```
name              'webserver'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures webserver'
long_description  'Installs/Configures webserver'
version           '0.2.0'
```


CHEF

COMMIT

GL: Commit Your Work

> cd ~/cookbooks/webserver
> git add .
> git status
> git commit -m "Release version 0.2.0"

CHEF

CONCEPT

Development Workflow

Make Change → Test Change → Update the Version → Commit Changes

CHEF

Discussion

What is the major difference between a single-quoted string and a double-quoted string?

How are the details about the system available within a recipe?

How does the version number help convey information about the state of the cookbook?

CHEF

Q&A

What questions can we help you answer?

- Ohai
- Node Object
- Node Attributes
- String Interpolation
- Semantic Versions

CHEF

CHEF™

Desired State and Data

Extracting the Content for Clarity

Objectives

After completing this module, you should be able to

- Explain when to use a template resource
- Create a template file
- Use ERB tags to display node data in a template
- Define a template resource

Cleaner Recipes

In the last section we updated our cookbook to display information about our node.

We added this content to the file resource in the recipe.

CHEF

Webserver Recipe

~\cookbooks\webserver\recipes\default.rb

```
package 'httpd'

file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>

  <h2>IPADDRESS: #{node['ipaddress']}</h2>
  <h2>HOSTNAME: #{node['hostname']}</h2>
  <h2>MEMORY: #{node['memory']['total']}</h2>
  <h2>CPU: #{node['cpu']['0']['mhz']}</h2>
"
end

service 'httpd' do
  action [:enable, :start]
end
```


CHEF

Double Quotes Close Double Quotes

Double quoted strings are terminated by double quotes.

```
"<h1 style="color: red;">Hello, World!</h1>"
```


CONCEPT

Backslash

We can use double-quotes as long as we prefix them with a backslash.

```
"<h1 style=\"color: red;\">Hello, World!</h1>"
```

Backslash

Backslashes are reserved characters. So to use them you need to use a backslash.

```
"Root Path: \"
```


CONCEPT

Backslash

Backslashes are reserved characters. So to use them you need to use a backslash.

```
"Root Path: \\"
```

Unexpected Formatting

```
file '/etc/motd' do
  content 'This is the first line of the file.
           This is the second line. If I try and line it up...

Don't even think about pasting ASCII ART in here!
'
end
```

```
This is the first line of the file.
           This is the second line. If I try and line
it up...

Don't even think about pasting ASCII ART in here!
```


Copy Paste

This process is definitely error prone. Especially because a human has to edit the file again before it is deployed.

CONCEPT

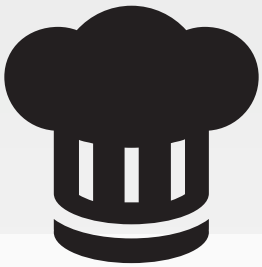
What We Need

We need the ability to store the data in another file, which is in the native format of the file we are writing out but that still allows us to insert ruby code...

...specifically, the node attributes we have defined.

CHEF

LAB



GL: Cleaner Recipes

Adding the node attributes to our recipes did make it harder to read.

OBJECTIVE:

- ❑ Decide which resource will help us address this issue

CHEF

DOCS

GL: Let's Check the Docs…

Use the file resource to manage files directly on a node.

Use the **cookbook_file** resource to copy a file from a cookbook's **/files** directory. Use the **template** resource to create a file based on a template in a cookbook's /templates directory. And use the **remote_file** resource to transfer a file to a node from a remote location.

https://docs.chef.io/resource_file.html

CHEF

DOCS

remote_file

Use the **remote_file** resource to transfer a file from a remote location using file specificity. This resource is similar to the file resource.

https://docs.chef.io/resource_remote_file.html

CHEF

DOCS

cookbook_file

Use the **cookbook_file** resource to transfer files from a sub-directory of COOKBOOK_NAME/files/ to a specified path located on a host that is running the chef-client.

https://docs.chef.io/resource_cookbook_file.html

CHEF

Demo: cookbook_file's Source Match Up

```
> tree cookbooks\webserver\files\default
files\default
└── index.html

0 directories, 1 file
```

```
cookbook_file '/var/www/html/index.html' do
  source 'index.html'
end
```


CHEF

DOCS

Template

A cookbook template is an Embedded Ruby (ERB) template that is used to generate files … Templates may contain Ruby expressions and statements and are a great way to...

Use the template resource to add cookbook templates to recipes; place the corresponding Embedded Ruby (ERB) template in a cookbook's /templates directory.

https://docs.chef.io/resource_template.html

CHEF

Demo: Template File's Source Matches Up

```
> tree cookbooks\webserver\templates\default
templates\default
└── index.html.erb

0 directories, 1 file
```

```
template '/var/www/html/index.html' do
  source 'index.html.erb'
end
```


CHEF

DOCS

Template

To use a template, two things must happen:

1. A template resource must be added to a recipe
2. An Embedded Ruby (ERB) template must be added to a cookbook

https://docs.chef.io/resource_template.html#using-templates

CHEF

LAB

GL: Cleaner Webserver Recipe

Adding the node attributes to the index page did make it harder to read the recipe.

OBJECTIVE:

- ❑ Create a template with chef generate
- ❑ Define the contents of the ERB template
- ❑ Change the file resource to the template resource in the 'webserver' cookbook

CHEF

GL: What Can `chef generate` Do?

```
> chef generate --help
```

```
Usage: chef generate GENERATOR [options]

Available generators:
  app          Generate an application repo
  cookbook     Generate a single cookbook
  recipe       Generate a new recipe
  attribute    Generate an attributes file
  template     Generate a file template
  file         Generate a cookbook file
  lwrp         Generate a lightweight resource/provider
  repo         Generate a Chef policy repository
  policyfile   Generate a Policyfile for use with the install/push commands
(experimental)
```


GL: What Can `chef generate template` Do?

```
> chef generate template --help
```

```
Usage: chef generate template [path/to/cookbook] NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright holder - defaults
to 'The Authors'
    -m, --email EMAIL                Email address of the author - defaults to
...
    -a, --generator-arg KEY=VALUE    Use to set arbitrary attribute KEY to
VALUE in the
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 -
defaults to
    -s, --source SOURCE_FILE         Copy content from SOURCE_FILE
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the
code_generator
        --generator-cookbook
```


GL: Use `chef` to Generate a Template

```
> cd ~
> chef generate template cookbooks\webserver index.html
```

```
Recipe: code_generator::template
  * directory[cookbooks/webserver/templates/default] action create
    - create new directory cookbooks/webserver/templates/default
  * template[cookbooks/webserver/templates/index.html.erb] action create
    - create new file cookbooks/webserver/templates/index.html.erb
    - update content in file cookbooks/webserver/templates/index.html.erb from
none to e3b0c4
    (diff output suppressed by config)
```

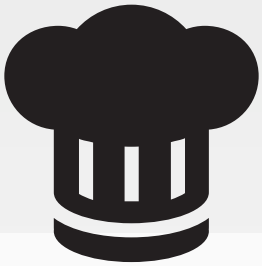

GL: Lets Look at the Template File

```
> tree /f cookbooks\webserver\templates
```

```
cookbooks/webserver/templates/
└── default
└── index.html.erb

1 directory, 1 file
```


LAB

Cleaner Recipes



Adding the node attributes to the default page did make it harder to read the recipe.

OBJECTIVE:

- ✓ Create a template with chef generate
- ❑ Define the contents of the ERB template
- ❑ Change the file resource to the template resource in the 'webserver' cookbook

CHEF

CONCEPT

ERB

An Embedded Ruby (ERB) template allows Ruby code to be embedded inside a text file within specially formatted tags.

Ruby code can be embedded using expressions and statements.

https://docs.chef.io/templates.html#variables

CHEF

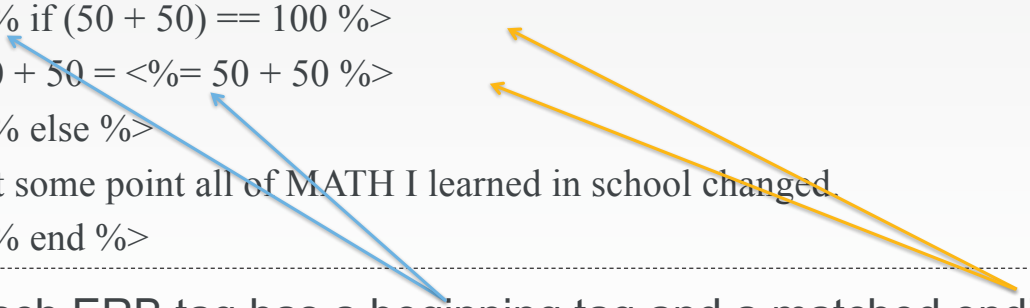
Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Each ERB tag has a beginning tag and a matched ending tag.

Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```



Each ERB tag has a beginning tag and a matched ending tag.

Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Each ERB tag has a beginning tag and a matched ending tag.

Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Executes the ruby code within the brackets and do not display the result.

Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Executes the ruby code within the brackets and display the results.

CONCEPT

The Angry Squid

<%=

GL: Move Our Source to the Template

~\cookbooks\webserver\templates\index.html.erb

```
<h1>Hello, world!</h1>
<h2>IPADDRESS: #{node['ipaddress']}</h2>
<h2>HOSTNAME: #{node['hostname']}</h2>
<h2>MEMORY: #{node['memory']['total']}</h2>
<h2>CPU: #{node['cpu']['0']['mhz']}</h2>
```

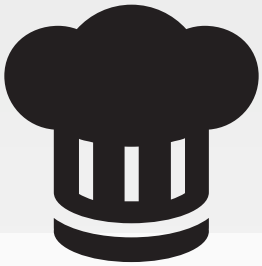

GL: Replace String Interpolation with ERB

~\cookbooks\webserver\templates\index.html.erb

```
<h1>Hello, world!</h1>
<h2>IPADDRESS: <%= node['ipaddress'] %></h2>
<h2>HOSTNAME: <%= node['hostname'] %></h2>
<h2>MEMORY: <%= node['memory']['total'] %></h2>
<h2>CPU: <%= node['cpu']['0']['mhz'] %></h2>
```


LAB

Cleaner Recipes



Adding the node attributes to the default page did make it harder to read the recipe.

OBJECTIVE:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ❑ Change the file resource to the template resource in the 'webserver' cookbook

CHEF

GL: Remove the Existing Content Attribute

~\cookbooks\webserver\recipes\default.rb

```
package 'httpd'

file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>

  <h2>IPADDRESS: #{node['ipaddress']}</h2>
  <h2>HOSTNAME: #{node['hostname']}</h2>
  <h2>MEMORY: #{node['memory']['total']}</h2>
  <h2>CPU: #{node['cpu']['0']['mhz']}</h2>
"
end

service 'httpd' do
  action [ :enable, :start ]
end
```


CHEF

GL: Change File Resource to a Template Resource

~\cookbooks\webserver\recipes\default.rb

```
package 'httpd'

template '/var/www/html/index.html' do
end

service 'httpd' do
  action [ :enable, :start ]
end
```


CHEF

What to Specify as the Source?

~\cookbooks\webserver\recipes\server.rb

```
package 'httpd'

template '/var/www/html/index.html' do
  source '????????????'
end

service 'httpd' do
  action [ :enable, :start ]
end
```


CHEF

GL: Viewing the Partial Path to the Template

```
> tree cookbooks\webserver\templates
```

```
cookbooks/webserver/templates
└── default
└── index.html.erb

1 directories, 1 file
```

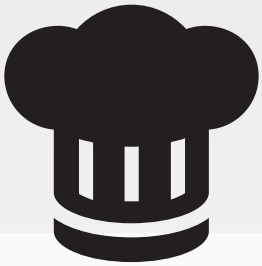

GL: Update the Source Attribute

~\cookbooks\webserver\recipes\server.rb

```
package 'httpd'

template '/var/www/html/index.html' do
  source 'index.html.erb'
end

service 'httpd' do
  action [ :enable, :start ]
end
```


LAB

Cleaner Recipes



Adding the node attributes to the default page did make it harder to read the recipe.

OBJECTIVE:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource in the 'webserver' cookbook

CHEF

EXERCISE

Lab: Update the Version

- ❑ Use kitchen test on the "webserver" cookbook
- ❑ Update the "webserver" cookbook's version for this patch
- ❑ Commit the changes

CHEF

Lab: Converge and Verify the Test Instance

```
> cd ~\cookbooks\webserver
> kitchen converge
> kitchen verify
```

```
...
-----> Starting Kitchen (v1.13.2)
-----> Converging <default-centos-6>...
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 5.1.0...
       Removing non-cookbook files before transfer
       Preparing validation.pem
       Preparing client.rb
-----> Chef Omnibus installation detected (install only if missing)
```

 233

CHEF

Lab: Update the Cookbook's Patch Number

~\cookbooks\webserver\metadata.rb

```
name              'webserver'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures webserver'
long_description  'Installs/Configures webserver'
version           '0.2.1'
```


234 2-

CHEF

COMMIT

Lab: Commit the Changes

> cd ~\cookbooks\webserver
> git add .
> git status
> git commit -m "Update default recipe to use template"

CHEF

EXERCISE

Lab: Update the Version

- ✓ Use kitchen test on the ”webserver" cookbook
- ✓ Update the ”webserver" cookbook's version for this patch
- ✓ Commit the changes

CHEF

Discussion

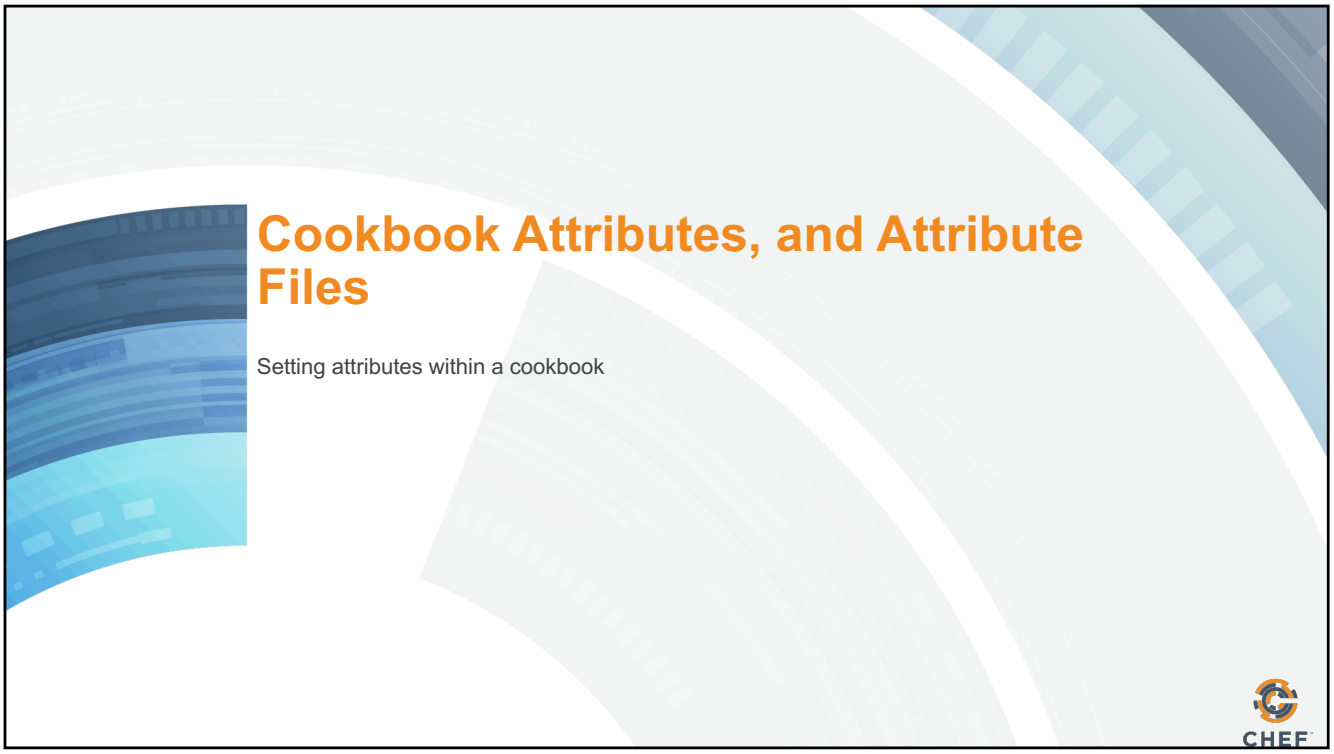
What is the benefit of using a template over defining the content within a recipe? What are the drawbacks?

What do each of the ERB tags accomplish?

Q&A

What questions can we help you answer?

- Resources (file, cookbook_file, template, and remote_file)
- Templates
- ERB



Objectives

After completing this module, you should be able to

- Explain where cookbook attributes reside
- Set attributes from within a cookbook
- Use Kitchen to converge and test the webserver cookbook

CONCEPT

Reconfigure Apache

Apache is configured to listen on port 80 in the file `/etc/httpd/conf/httpd.conf`

We will create a template for this file in the apache cookbook and change the value to 8080 via an attribute.

CONCEPT

Attribute Files

The Node Object contains many automatic attributes generate by OHAI.

You can also maintain attributes within a cookbook.

These are like variables or parameters for your cookbook and allow recipes to be data driven.

CONCEPT

Best Practices

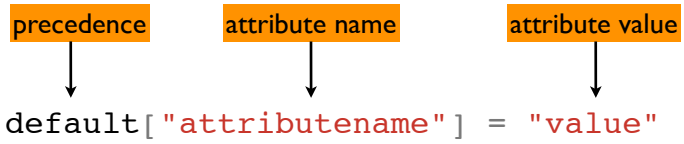
- Well-written cookbooks change behavior based on attributes.
- Ideally, you don't have to modify the contents of a cookbook to use it for your specific use case.
- Look at the attributes directory for things you can override through roles to affect behavior of the cookbook.
- Of course, well written cookbooks have sane defaults, and a README to describe all this.

Setting Attributes in Attribute Files

Cookbook attributes are set in the attributes file

`./cookbooks/<cookbook>/attributes/default.rb`

Format is:



We'll look at precedence later.

EXERCISE

GL: Update

So we want Apache to serve content of port tcp/8080

Objective:

- ❑ Reconfigure Apache to serve content of port tcp/8080
- ❑ Test the change with Test Kitchen

GL: Bump the Cookbook Version Number

~\cookbooks\webserver\metadata.rb

```
name              'webserver'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures webserver'
long_description  'Installs/Configures webserver'
version           '0.3.0'
```


CHEF

GL: Generate the attributes File

```
> cd ~
> chef generate attribute cookbooks\webserver default
```

```
Compiling Cookbooks...
Recipe: code_generator::attribute
  * directory[cookbooks/webserver/attributes] action create
    - create new directory cookbooks/webserver/attributes
  * template[cookbooks/apache/attributes/default.rb] action create
    - create new file cookbooks/webserver/attributes/default.rb
    - update content in file
cookbooks/webserver/attributes/default.rb from none to e3b0c4
    (diff output suppressed by config)
```

CHEF

GL: Set the Port Value as an Attribute

~\cookbooks\webserver\attributes\default.rb

```
default['webserver']['port'] = 8080
```

Its good practice to include the name of the cookbook in the attribute name – helps trace where the value is set, although this is not enforced.

GL: Update the webserver::default Recipe

~\cookbooks\webserver\recipes\default.rb

```
...
template '/var/www/html/index.html' do
  source 'index.html.erb'
end

template '/etc/httpd/conf/httpd.conf' do
  action :create
  source 'httpd.conf.erb'
  notifies :restart, 'service[httpd]'
end

service 'httpd' do
  action [ :enable, :start ]
end`
```


GL: Generate the Template file

```
> cd ~
> chef generate template cookbooks\webserver httpd.conf
```

```
Compiling Cookbooks...
Recipe: code_generator::template
  * directory[cookbooks/webserver/templates/default] action create (up to
date)
  * template[cookbooks/webserver/templates/default/httpd.conf.erb] action
create
    - create new file cookbooks/webserver/templates/default/httpd.conf.erb
    - update content in file
cookbooks/webserver/templates/default/httpd.conf.erb from none to e3b0c4
    (diff output suppressed by config)
```

GL: Add the 'port' Variable to the Template

~\cookbooks\apache\templates\default\httpd.conf.erb

```
...
MaxSpareThreads      75
ThreadsPerChild      25
MaxRequestsPerChild  0
</IfModule>

~~Listen 80~~
Listen <%= node['webserver']['port'] %>

LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
...
```

Go to: http://bit.ly/1Hdx7E1
...click the Raw button, and copy/paste the contents of the file

After the copy/paste, modify the line (line 27) as shown above.

GL: Update the webserver cookbook tests

~\cookbooks\webserver\test\smoke\default\default_test.rb

```
unless os.windows?
  describe user('root') do
    it { should exist }
  end
end

describe port(8080) do
  it { should be_listening }
end

describe command('curl localhost:8080') do
  its('stdout') { should match('Hello, world') }
end
```


CHEF

GL: Converge and Verify the test instance

```
> cd ~\cookbooks\webserver
> kitchen converge
> kitchen verify
```

```
...
-----> Starting Kitchen (v1.13.2)
-----> Converging <default-centos-6>...
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 5.1.0...
       Removing non-cookbook files before transfer
       Preparing validation.pem
       Preparing client.rb
-----> Chef Omnibus installation detected (install only if missing)
```

CHEF

Discussion

Attributes are like parameters to your cookbook – no hard-coded values in recipes or templates.

Can you imagine in complex topologies, where you could have multiple levels of dependencies between cookbooks – berkself handles all of that out the box?

Q&A

What questions can we answer for you?


CHEF™


Chef Automate
Workflow Overview
Defining the Chef Workflow Pipeline
CHEF

Objectives

After completing this module, you should be able to:

- Explain Continuous Integration and Deployment.
- Describe the capabilities of Chef Automate Workflow.
- List the Workflow pipeline stages and phases.

CHEF

CONCEPT

Continuous Integration and Deployment

Continuous Integration is the practice of testing each change done to your codebase automatically and as early as possible.

Continuous Deployment follows the testing that happens during Continuous Integration and pushes changes to a staging or production system.

This makes sure a version of your code is accessible at all times.

CHEF

CONCEPT

Continuous Integration and Deployment

Continuous Integration requires developers to integrate code into a shared repository several times a day.

Each check-in is then verified by an automated build, allowing teams to detect problems early.

By integrating regularly, you can detect errors quickly, and locate them more easily.

Chef Automate Workflow is the platform that provides continuous integration and deployment.

CHEF

CONCEPT

Chef Automate Workflow and git

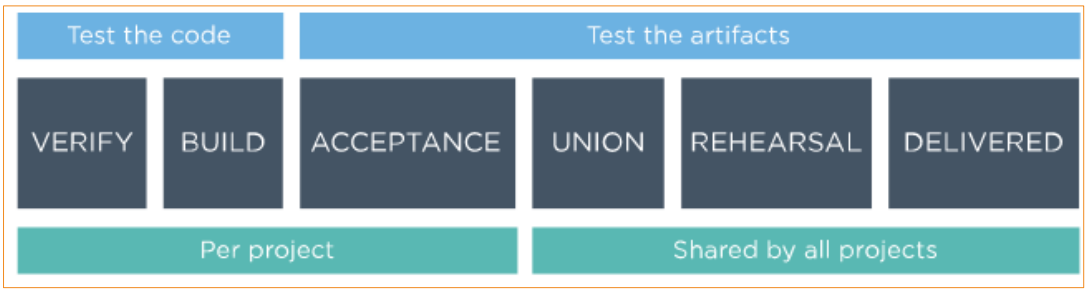
Workflow relies on git and uses its lightweight feature branches as the mechanism for handling changes before they merge, as well as its ability to perform merges automatically.

Each Workflow pipeline has a designated target branch into which it will merge approved changes.

CHEF

Chef Automate Workflow Pipeline Stages

A Workflow pipeline is a series of automated and manual quality gates that take software changes from development to delivery.

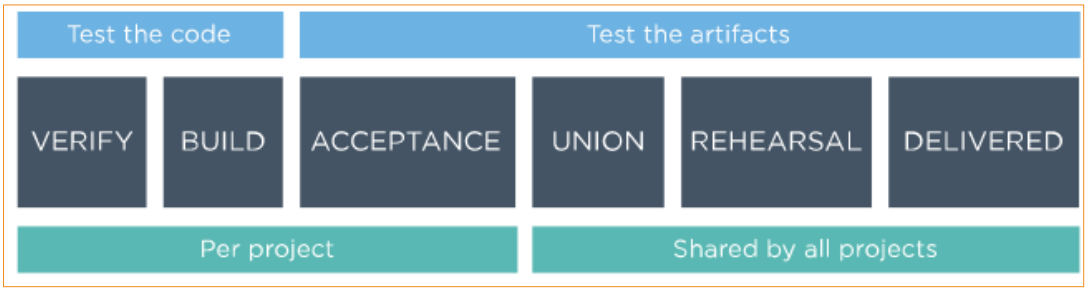


CHEF

Chef Automate Workflow Pipeline Stages

The Workflow pipeline is made up of six stages: Verify, Build, Acceptance, Union, Rehearsal, and Delivered.

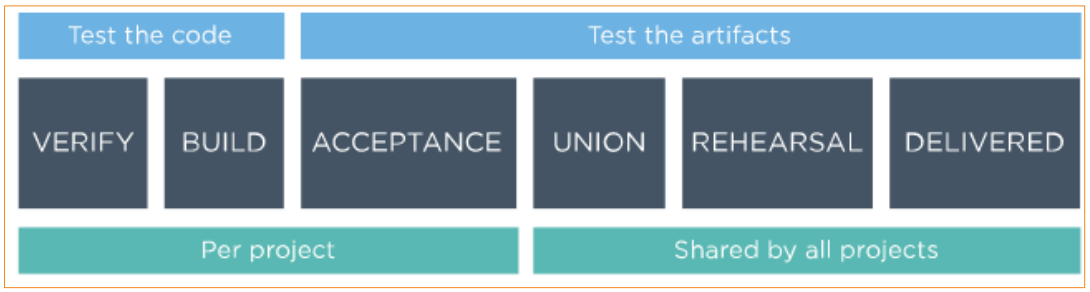
Each stage consists of phases that perform a particular task, such as running some type of test.



CHEF

Chef Automate Workflow Pipeline Stages

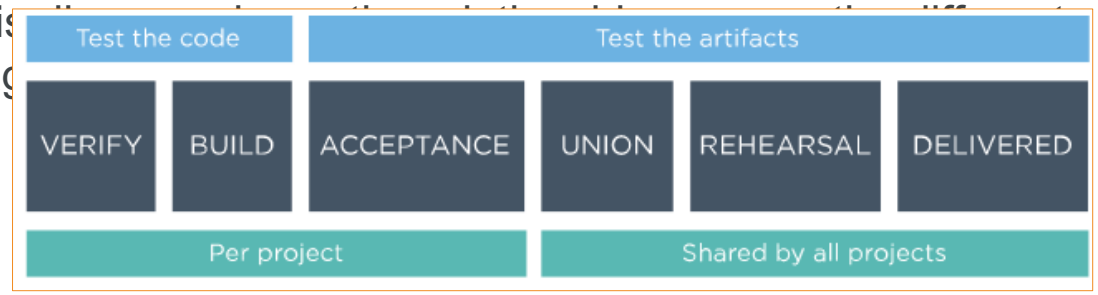
One way to think about the stages is whether the set of potentially releasable artifacts (a piece of code, a cookbook, etc.) has been produced or not. The pipeline creates these artifacts at the end of the Build stage.



Chef Automate Workflow Pipeline Stages

The remaining stages of the pipeline focus on gaining confidence in those artifacts. Another way to understand the stages is by whether they are isolated at the project level or shared across the system.

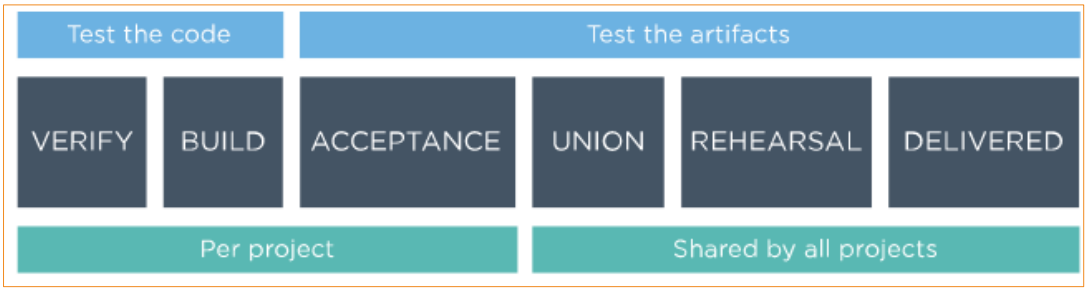
This
stag



Chef Automate Workflow Pipeline Stages Summary

The Verify and Build stages perform tests on the source code.

The Acceptance, Union, Rehearsal and Delivered stages test potentially releasable artifacts. The Union, Rehearsal, and Delivered stages constitute the shared Workflow pipeline.

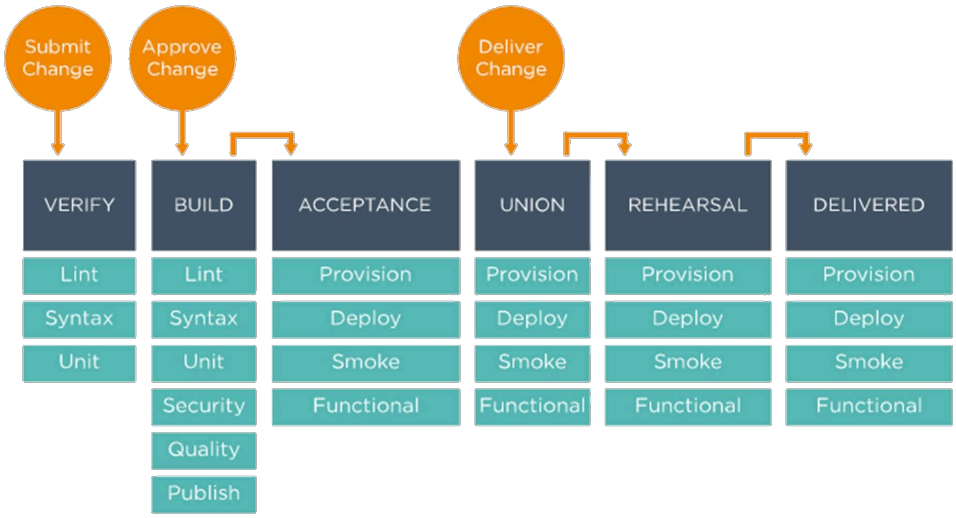


CHEF

Chef Automate Workflow Pipeline Phases

The tests within each stage are organized into phases.

The turquoise boxes in this example represent

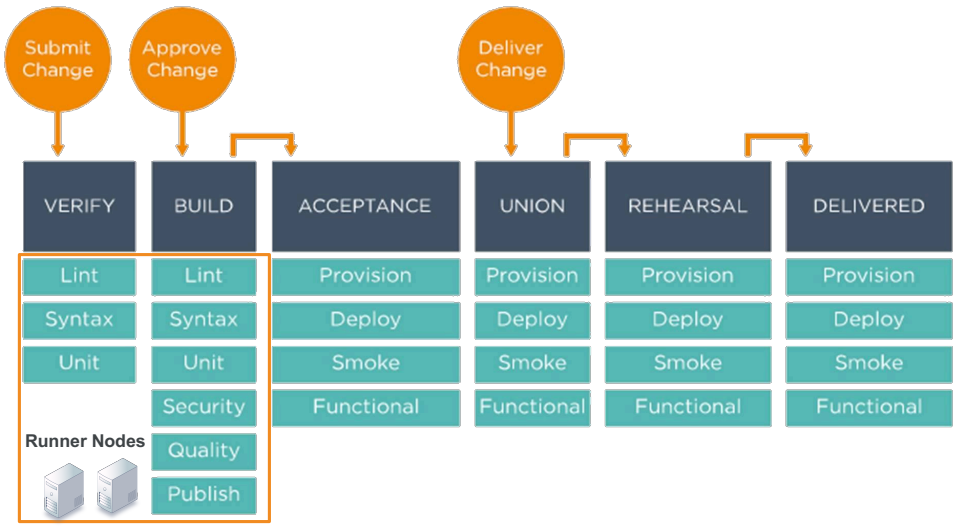


CHEF

Chef Workflow Pipeline Stages, Phases and Hardware

Runner nodes run the tests shown in turquoise under the Verify and Build stages.

Each of our lab systems has three to five

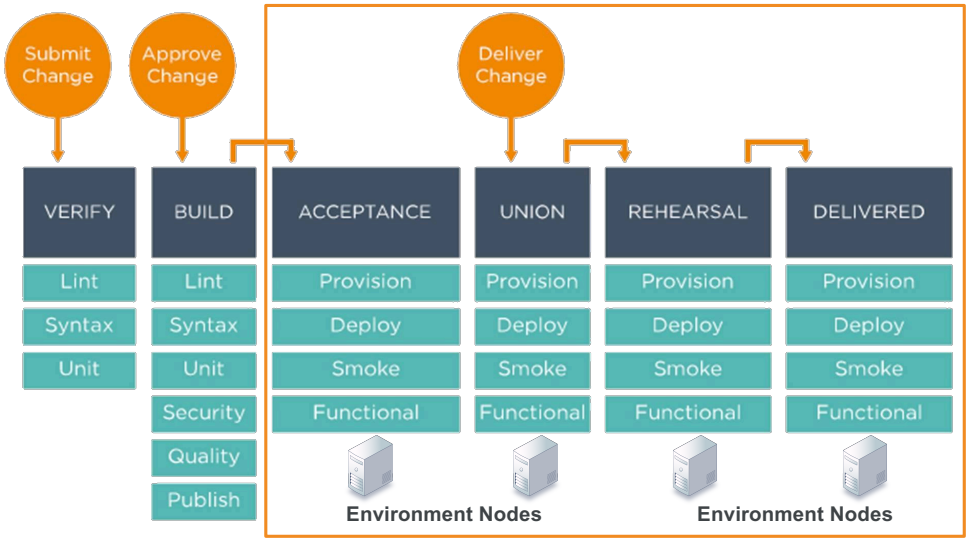


CHEF

Chef Workflow Pipeline Stages, Phases and Hardware

Environment nodes run the Acceptance, Union, Rehearsal, and Delivered phases.

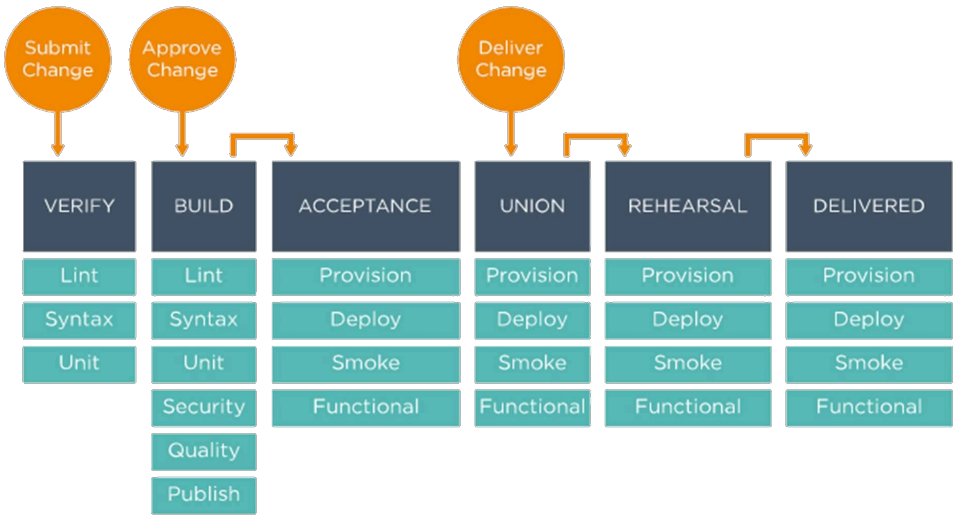
This is done using Chef Server and



CHEF

Chef Workflow Pipeline Summary

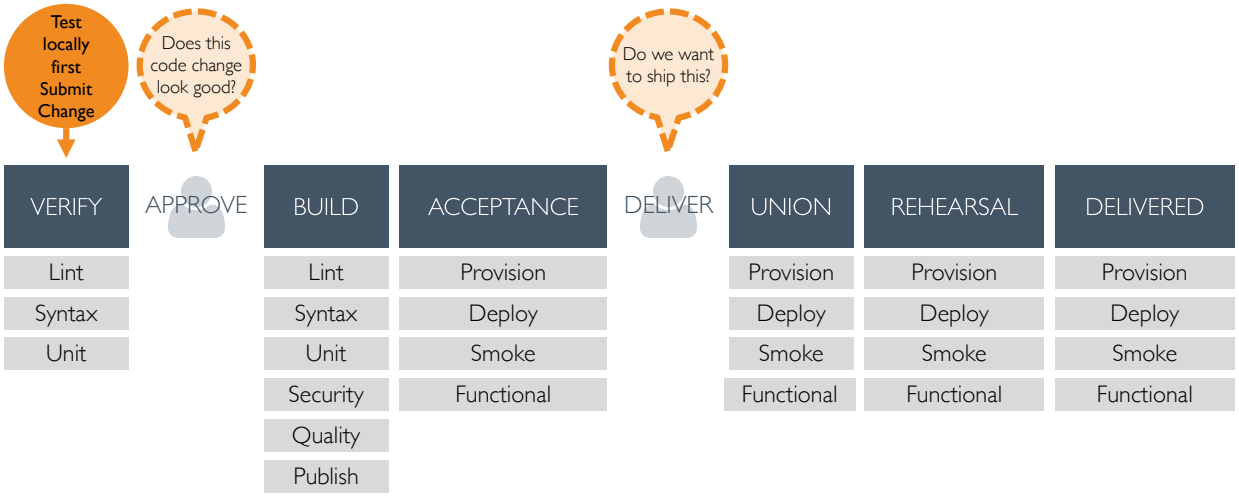
As changes flow through the Workflow pipeline, they are tested in a series of runtime environments that are increasingly similar to the final runtime



CHEF

Chef Workflow Approval Gates

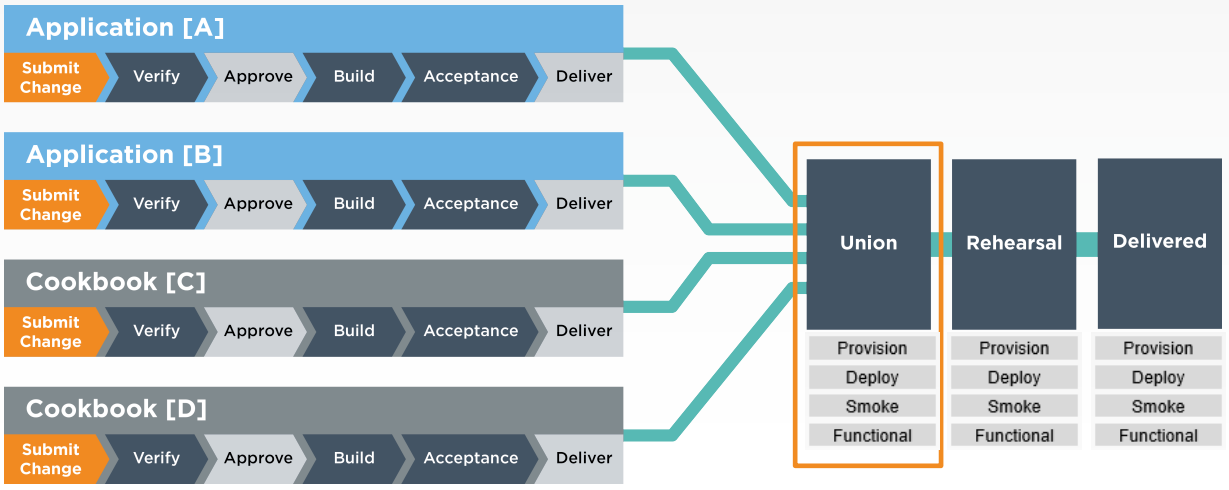
Workflow includes two explicit review and approval gates.



CHEF

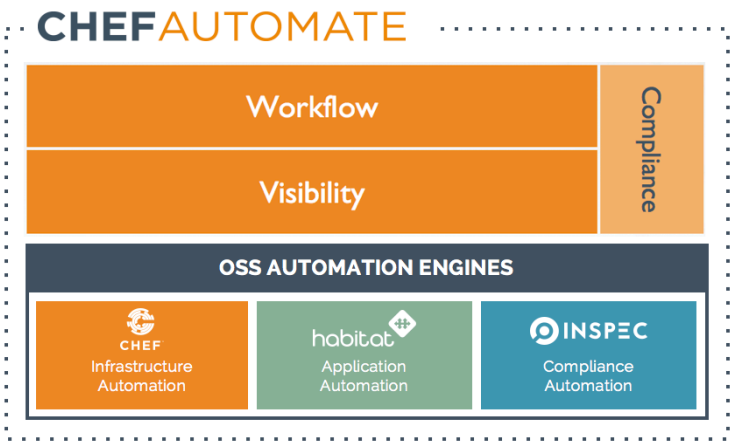
Shared Workflow

Workflow's pipeline is **shared** across projects and teams



CHEF

Chef Provides a Proven Approach to DevOps



CHEF

Q&A

What questions can we answer for you?

- Continuous Integration and Deployment.
- Capabilities of Chef Workflow.
- Workflow pipeline stages and phases.

CHEF

CHEF™

Creating and Submitting a Project

Submitting a Project from the Verify Stage to Delivered Stage

Objectives

After completing this module, you should be able to:

- Generate your cookbook and iterate the code.
- Use Kitchen to test your code
- Use the `delivery init` and `delivery review` commands to submit a project.
- Approve the change.
- Deliver the change.
- Integrate the change to your local master branch.

CONCEPT

Chef Automate Workflow Projects

Chef Automate workflow uses *projects* to organize work across multiple teams.

Each project has its own Git repository. Chef Automate workflow can host the Git repository for you or you can connect Chef Automate workflow to an existing project, such as one on GitHub.

In this module, you'll get starter code from GitHub but you'll host your project in Chef Automate's Git repository.

CHEF

CONCEPT

Chef Automate Set Up

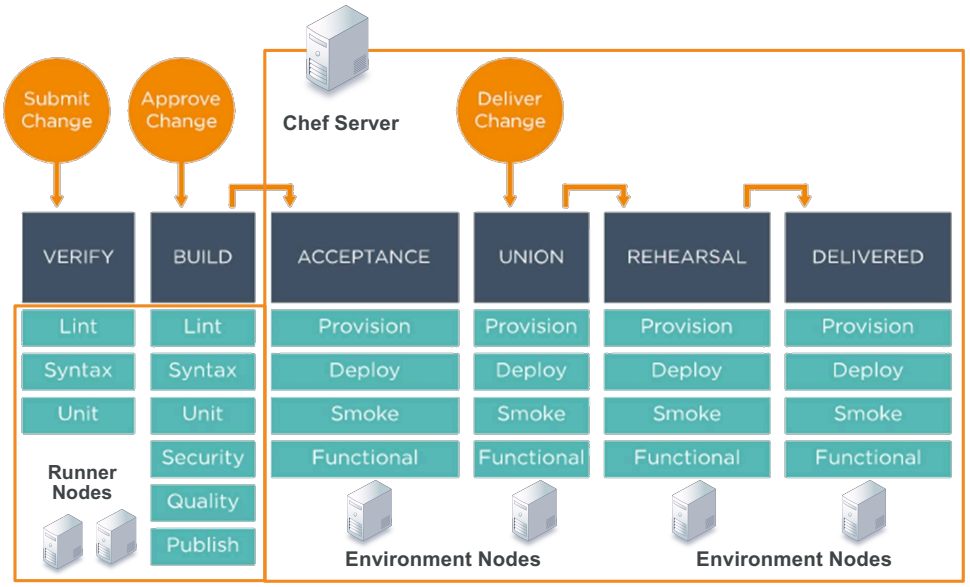
We've already created your Chef Automate cluster, including your virtual workstation. That cluster includes:

- ❑ Virtual Workstation
- ❑ Chef Automate server
- ❑ Chef server
- ❑ Build node 1
- ❑ Build node 2
- ❑ Build node 3
- ❑ Acceptance node
- ❑ Union node
- ❑ Rehearsal node
- ❑ Delivered node
- ❑ Compliance server*
  - ❑ *Not used in this course but is used in the Chef Automate compliance course.

CHEF

Your Chef Automate Cluster Lab System

You will use your Windows virtual workstation to interact with the Workflow server, which will interact with the other nodes.



CHEF

EXERCISE

Group Lab: Creating a Project and a Automate Workflow Pipeline

Objective:

- ❑ Create your project
- ❑ Initialize the project.
- ❑ View the Verify stage.

CHEF

Cookbooks Generate in This Course

Each two-person team in each lab system cluster will generate a new cookbook in this course.

For example:

Team 1 will generate and pair program on a cookbook

Team 2 will generate and pair program on a cookbook

Workstations Used in This Course

The instructor will provide you with your assigned workstations IP addresses and your team names (Team 1, Team 2, etc.).

Each two-person team will be provided with two workstations:

Team 1:
One for the Submitter. For example, workstation-1.

One for the Reviewer. For example, workstation-2.

Team 2:
One for the Submitter. For example, workstation-3.

Workstations Used in This Course

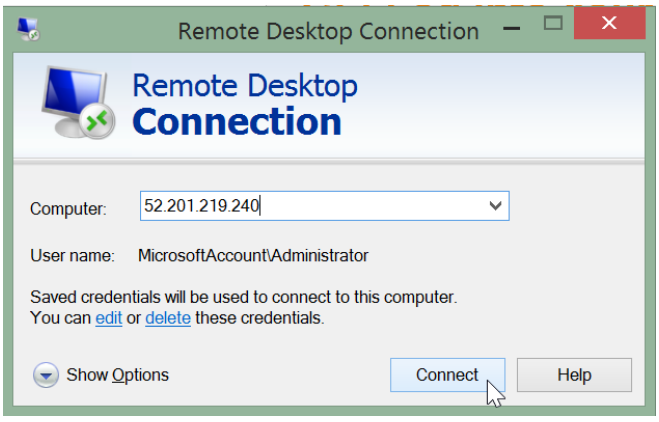
Each two-person team should now decide who will be the project Submitter, and who will be the project Reviewer. *You can change who the Submitter and Reviewers are later as well.*

The project Submitter will perform more tasks than the Reviewer but the Reviewer will pair program with the Submitter as well as performing the Reviewer tasks.

GL: Log into Your Virtual Workstation

The instructor will now provide you with your assigned workstations IP addresses.

Use your local Remote Desktop Connection and your assigned IP address to connect to your Windows Virtual workstation.

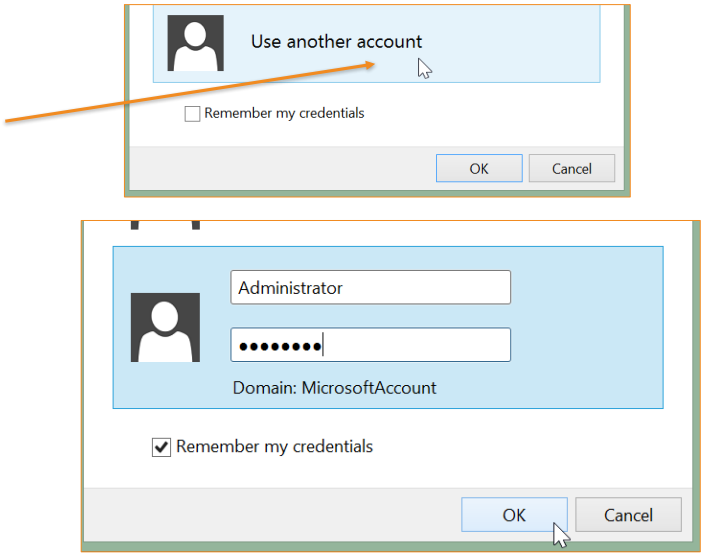


GL: Log into Your Virtual Workstation

On the resulting log in prompt, select **Use another account** and then log in with these credentials:

Username: `chef`

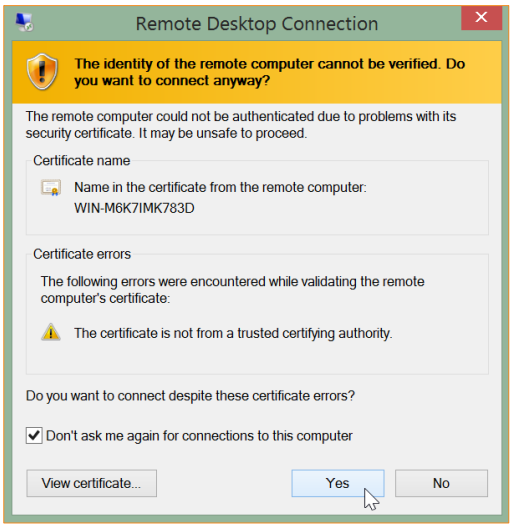
Password: `RL9@T40BTmXh`



GL: Log into Your Virtual Workstation

From the resulting Security Certificate warning, click **Don't ask me again**... and then click **Yes**.

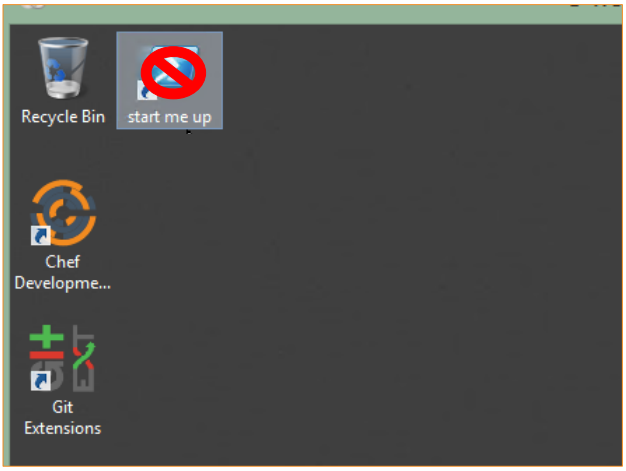
Note: The first time you log into your assigned workstation, you may need to wait a few minutes for the desktop to populate



GL: Important: Do Not Click This

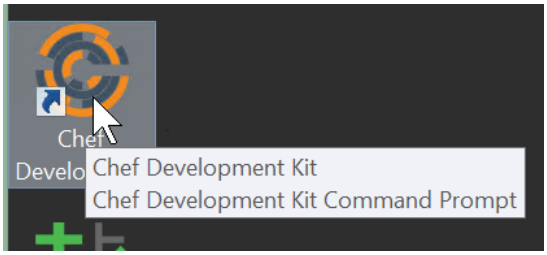
Important: Please do not click the "start me up" icon.

It is there only for the instructor's use.



GL: Launch the ChefDK CLI (Delivery CLI)

From the top-left of the virtual workstation's desktop, double-click the special **Chef Development Kit command prompt** icon to start a special PowerShell instance. This CLI includes the Automate functionality of what is called the Delivery CLI.



GL: Important Note About Who is Driving

The following tasks must be performed by only one member of your assigned team. We call that team member the **Submitter**.

The slides will indicate when the other team member, the **Reviewer**, has a task to perform. Otherwise, the **Reviewer** can observe and advise the **Submitter's** tasks and vice versa.

GL: Submitter: Let's Create a Cookbook

```
> cd ~
> cd cookbooks
> chef generate cookbook yourusecase
```

```
Generating cookbook yourusecase
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master

Your cookbook is ready. To setup the pipeline, type `cd yourusecase`, then run
`delivery init`
```


GL: Submitter: Move to `yourusecase` cookbook repo

```
C:\Users\chef\cookbooks> cd yourusecase
C:\Users\chef\cookbooks\yourusecase [master ≡]> ls
```

```
Directory: C:\Users\chef\cookbooks\automate-workflow-course-repo

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          10/5/2016   3:42 PM            base_web-1
d----          10/5/2016   3:42 PM            base_web-2
d----          10/5/2016   3:42 PM            base_web-3
d----          10/5/2016   3:42 PM            base_web-4
d----          10/5/2016   3:42 PM            site_config-1
d----          10/5/2016   3:42 PM            site_config-2
d----          10/5/2016   3:42 PM            site_config-3
d----          10/5/2016   3:42 PM            site_config-4
-a---          10/5/2016   3:42 PM         67 README.md
```

GL: Submitter: Run VS Code to Edit the Cookbook

```
C:\Users\chef\cookbooks\yourusecase [master ≡]> code .
```

GL: Submitter: Update the Workflow config

~/cookbooks/yourcookbook/.delivery/config.json

```
{
  "version": "2",
  "build_cookbook": {
    "name": "build_cookbook",
    "path": ".delivery/build_cookbook"
  },
  "delivery-truck": {
    "publish": {
      "chef_server": true
    }
  },
  "job_dispatch": {
    "version": "v1"
  },
  "skip_phases": [
    "lint",
    "unit",
    "quality",
    "security"
```

Grab this conf file from:
https://goo.gl/fXaL2f

CHEF

GL: Submitter: Run `git add .`

```
C:\Users\chef\cookbooks\yourusecase [master +0 ~2 -0 !]> git add .
C:\Users\chef\cookbooks\yourusecase [master +0 ~2 -0 ~]> git commit -m "Bootstrapping my pipeline"
```

```
[master 1577912] Initial commit of my cookbook
 2 files changed, 3 insertions(+), 3 deletions(-)
```

CHEF

Automate Workflow Project Discussion (1 of 2)

There are a few ways to create a Chef Automate workflow project. In this course, you will use the **delivery init** command to create a project. This command:

- Creates a project in Chef Automate, which includes a new Git repository that's hosted on the Chef Automate server.
- Creates a default pipeline whose target branch is master.
- Initializes the master branch in Chef Automate's Git repo from the existing master branch that you just cloned (`delivery init` detects whether there is an existing Git repository)

Chef Automate Project Discussion (2 of 2)

delivery init:

- Creates a branch named initialize-delivery-pipeline, which is based off of master.
- Creates the .delivery directory and adds to it a build cookbook and a configuration file.

GL: Submitter: Run `delivery init` and Type the Password When Prompted

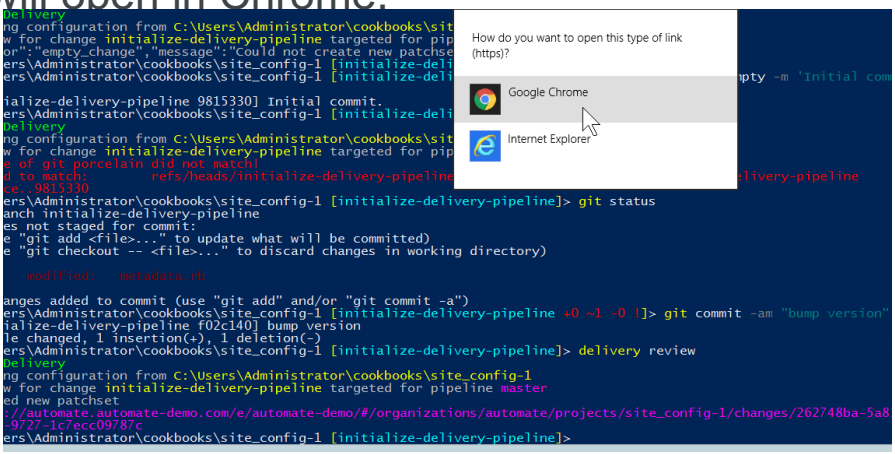
```
C:\Users\chef\cookbooks\yourusecase [initialize]> delivery init
```

```
Chef Delivery
Loading configuration from C:\Users\chef\cookbooks\yourusecase
Requesting Token
Delivery password:
token: JApxs61balm5tsYVY9jDQpE7GA+n+6x2ACL5tp8hKds=
saved API token to: C:\Users\chef\.delivery\api-tokens
Creating Delivery project...
  Delivery project named yourusecase was created. Review submitted to Delivery.

Your new Delivery project is ready!
```

CHEF

GL: Submitter: Initialize Your Project

When prompted, select Google Chrome and your project pipeline will open in Chrome.



CHEF

GL: Submitter: Type the Chef Automate Credentials

When prompted, type the Chef Automate server's credentials. They are:

User Name: **workstation-#** where **workstation-#** is the workstation name of the Submitter.

Password: **workstation!**

Also check the **Remember Me**

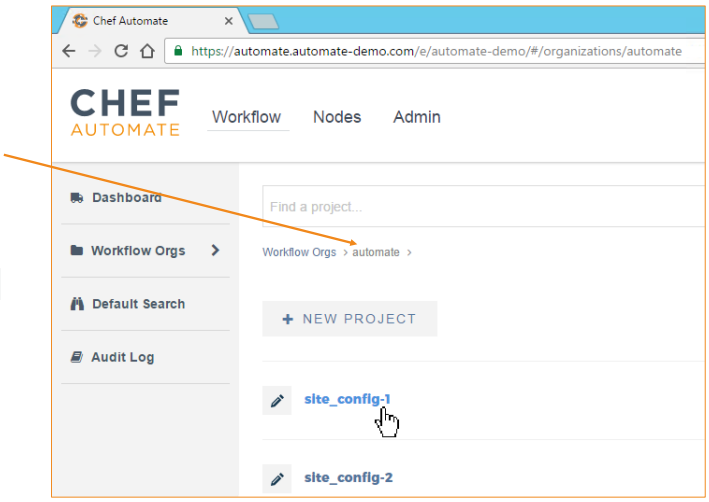
GL: Reviewer: Log into Chef Automate

Reviewer: From your virtual workstation, launch Google Chrome and point it to https://automate.automate-demo.com/e/automate-demo/#/organizations/automate

User Name: **workstation-#** where **workstation-#** is the workstation name of the Reviewer

GL: Reviewer: Navigate to your Project

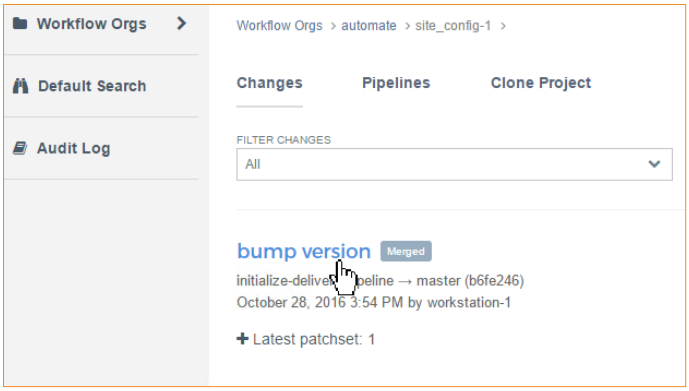
From this Workflow Orgs page you can see all the projects residing under the organization (**automate** organization) that we are all using in this course.



Reviewer: Click the project name your team is working on.

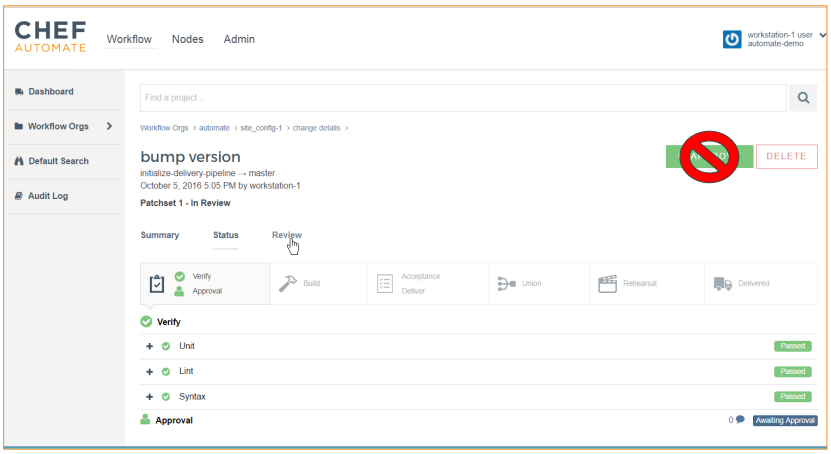
GL: Reviewer: Navigate to your Project

From the resulting window, click the **New pipeline verification commit** patchset link to proceed to the pipeline.



GL: Reviewer: Viewing the Verify Stage of the Pipeline

Your Chef Automate pipeline should display at this point and you should see the Verify stage running or that is has finished running.



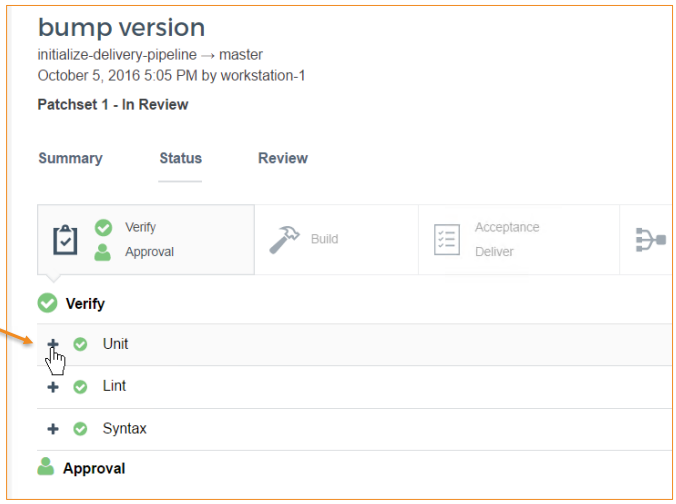
Phases:

- Unit tests run rspec.
- Lint tests run

GL: Reviewer: Viewing the Verify Stage of the Pipeline

From this view you could see the actual code that comprised each test by using the **+** (plus) links.

To do so, click the **+** that's to the right of the Unit test.

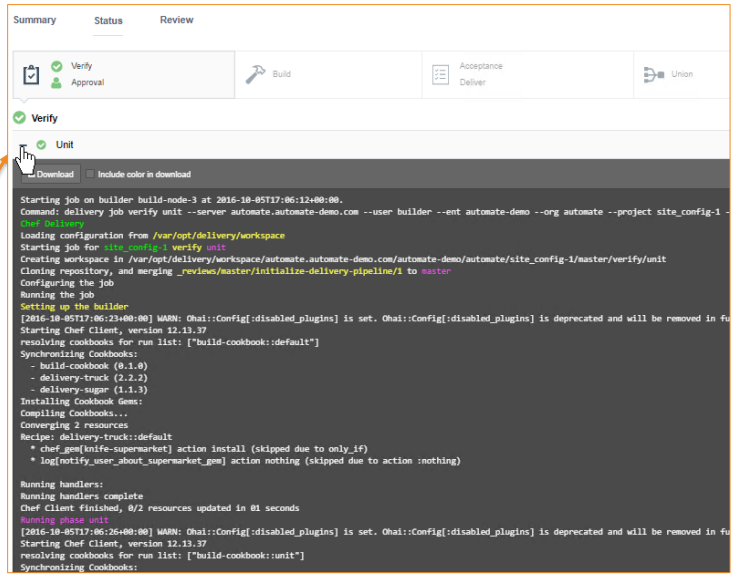


GL: Reviewer: Viewing the Verify Stage of the Pipeline

In the workplace you can view these test details from any of the + links.

You could also download the test details using the Download button.

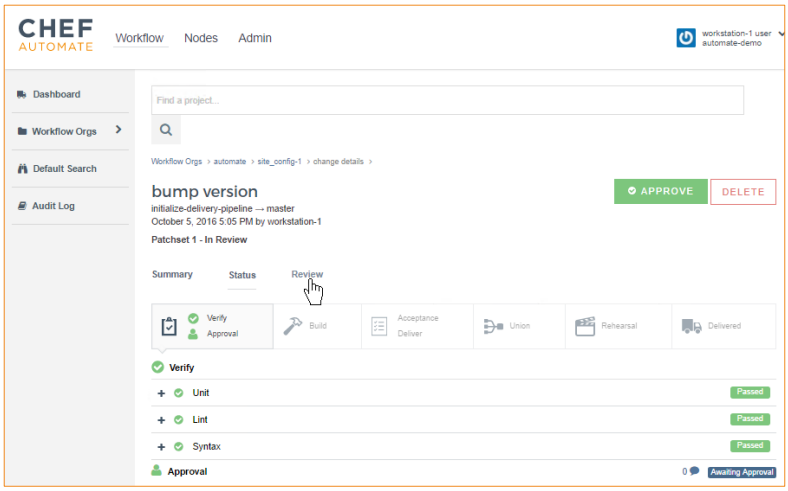
Click the - (minus)



GL: Reviewer: Review the Changes

Now that your Verify stage is complete, a manual step is required to move to the Build stage.

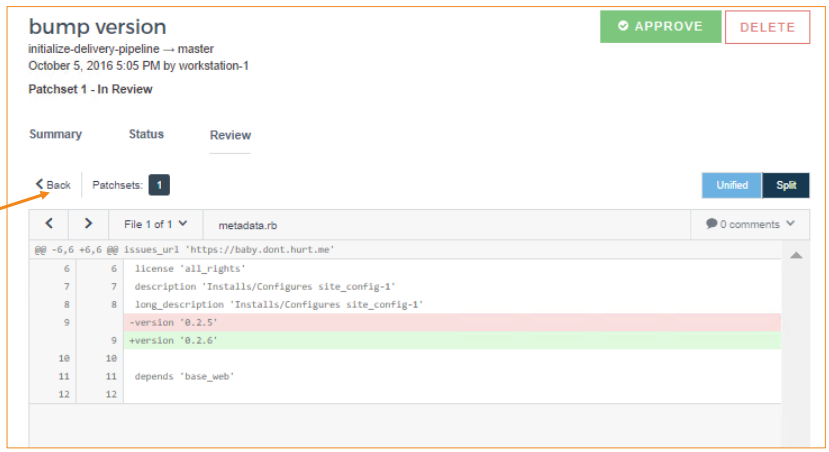
This is where your team decides whether the code can be merged into the master branch or more work is needed.



GL: Reviewer: Review the Changes

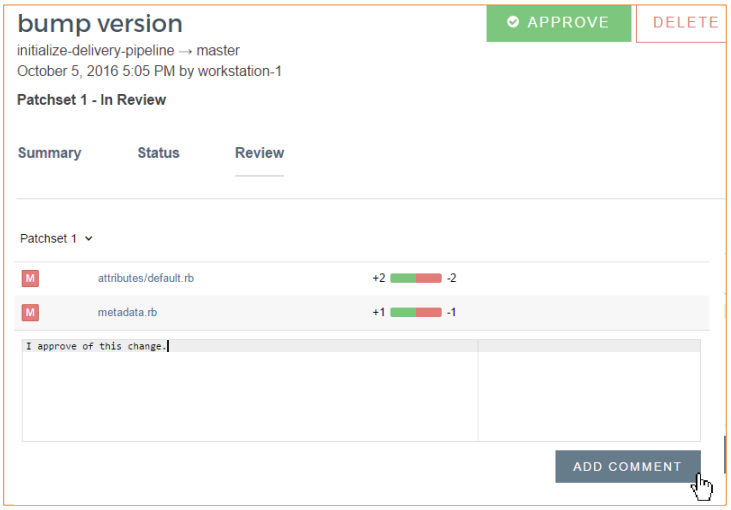
Now you can see the changes that are pending your approval.

Reviewer: Click the **< Back** link to back up.



GL: Reviewer: Review the Changes

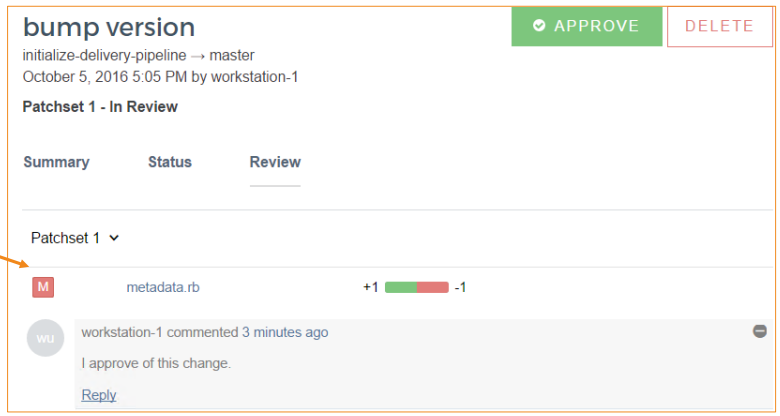
Reviewer: Type an approving comment in the field provided and then click the **Add Comment** button.



GL: Reviewer: Review the Changes

Notice that the comment can be replied to.

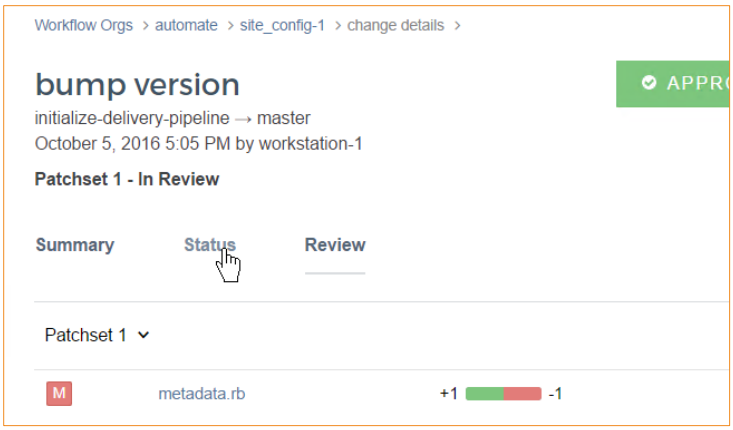
Depending on your change, you may see the M icon. This indicates that this is a modification to the project.



GL: Reviewer: Review the Changes

Reviewer: Click the **Status** link to return to the pipeline.

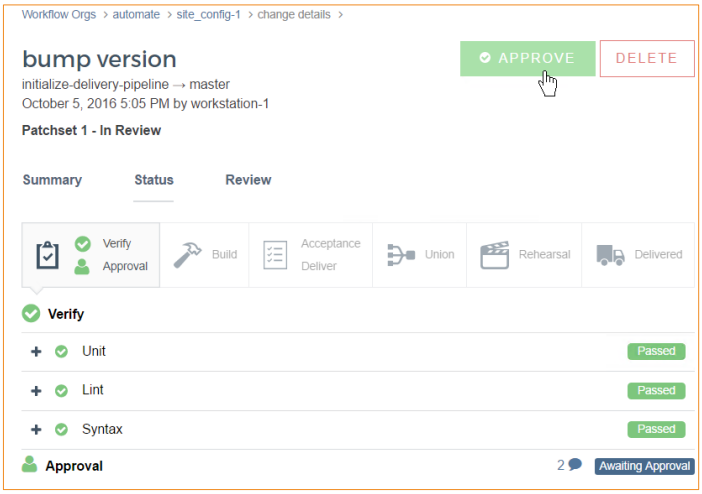
Reviewer: You can continue to view the progress of the project from your assigned virtual workstation if you like.



GL: Reviewer: Approve the Change

Since the reviewer has signed off of this change, we can approve the change.

Submitter: Click the **Approve** button to continue the workflow

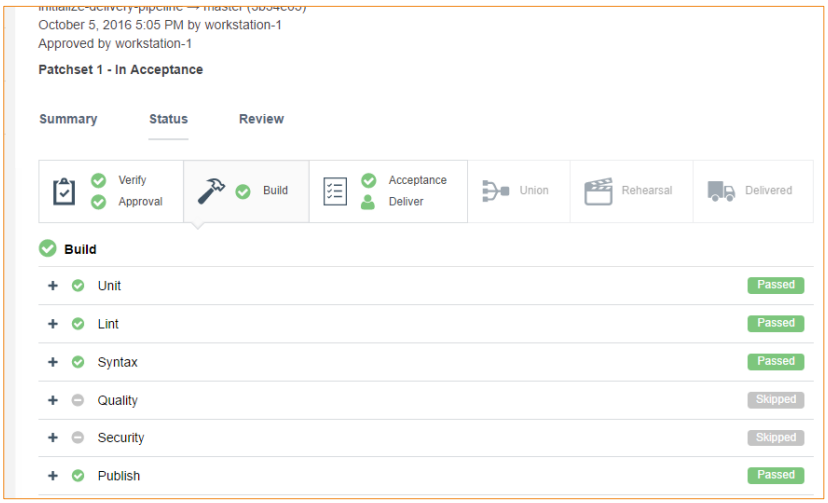


GL: Build Stage and Its Phases

The Build stage begins.

In a few moments you should see your project go through the Build stage phases:

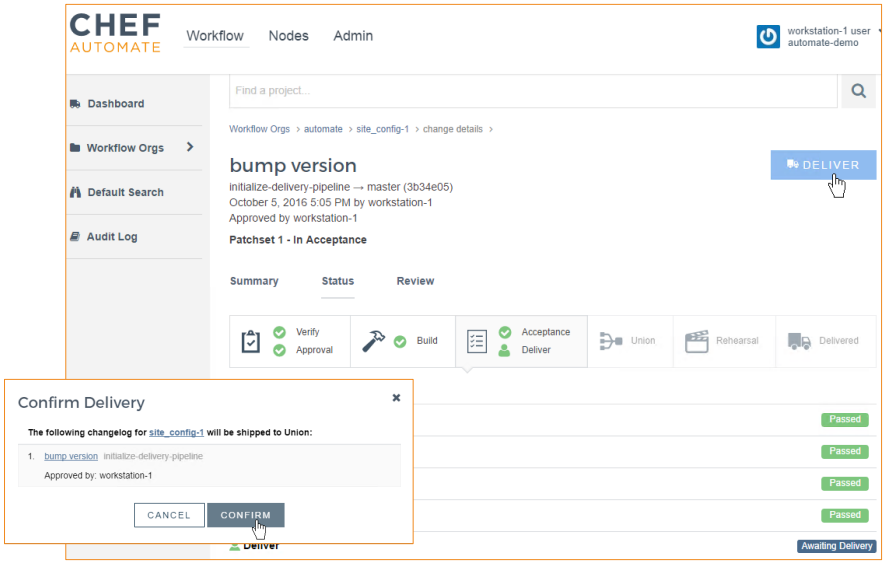
- Unit
- Lint
- Syntax
- Quality
- Security
- Publish



GL: Reviewer: Deliver the Artifact

Reviewer: Click the **Deliver** button and the resulting **Confirm** button.

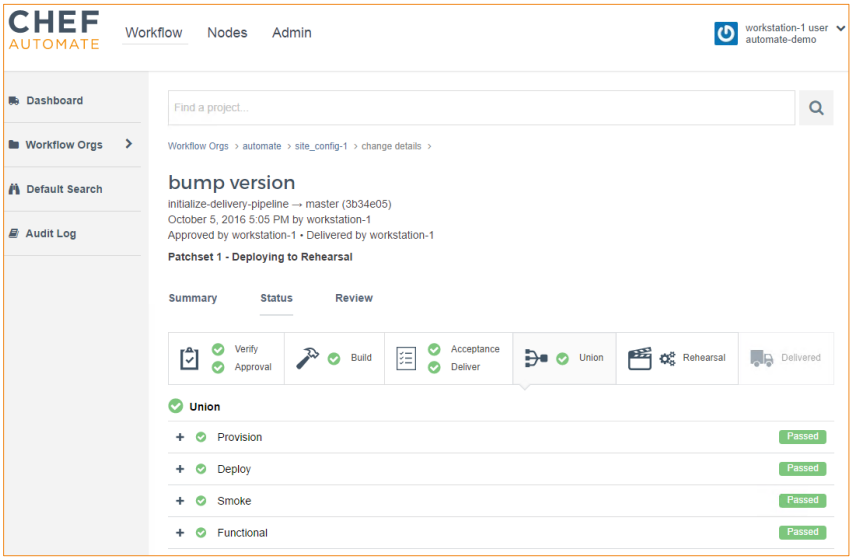
This action confirms that the artifact can be released. The artifact will move through the next three pipeline stages automatically.



GL: Deliver the Artifact

Here is the Union stage.

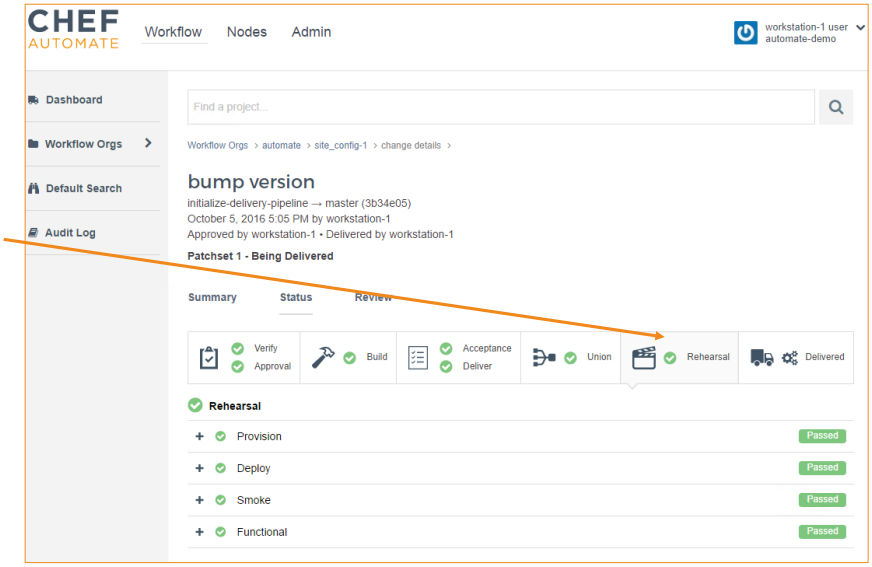
Note: You may have to wait for your project to start running through Union if another project was sent to Union before yours did. This is normal behavior.



GL: Deliver the Artifact

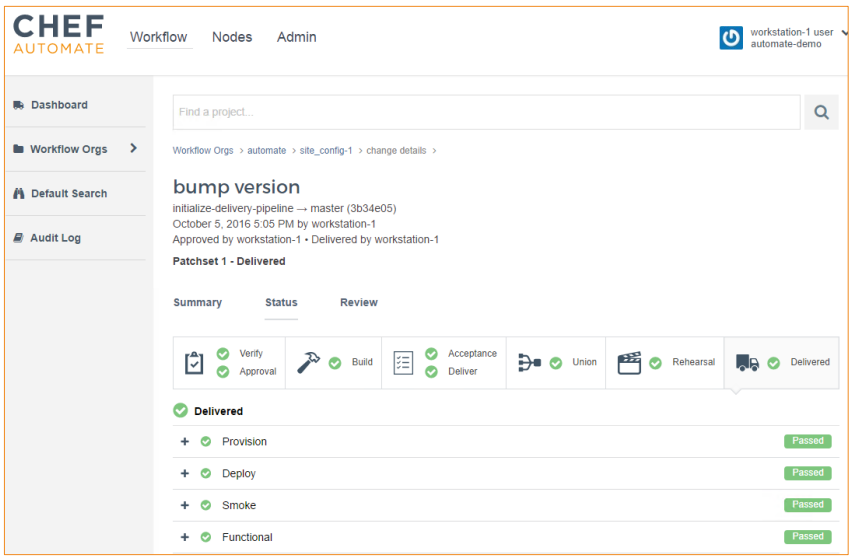
Here is the Rehearsal stage.

Note: You may need to click the Stage tabs if you want to see the tests in progress.



GL: Deliver the Artifact

And here is the conclusion of the Delivered stage.



GL: Integrate the Change to Your Local Master Branch

When you clicked Approve during the Verify stage, Chef Delivery merged the add-delivery-config branch into master on Delivery's Git server.

Next we'll need the Submitter to pull Delivery's updated master into your local master branch.

Be sure your project pipeline has completed the Delivered stage before running the next commands.

GL: Submitter: Integrate the Change to Your Local Master Branch

```
[initialize-delivery-pipeline]> git checkout master
```

```
Switched to branch 'master'
Your branch is up-to-date with 'delivery/master'.
```

GL: Submitter: Integrate the Change to Your Local Master Branch

```
\cookbooks\yourusecase [master ≡]> git pull --prune
```

```
From ssh://automate-demo@automate.automate-demo.com:8989/automate-demo/automate/site_config-1
 x [deleted]         (none)     -> delivery/_for/master/initialize-delivery-pipeline
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
   d01bece..3b34e05  master     -> delivery/master
Updating d01bece..3b34e05
Fast-forward
 metadata.rb | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

GL: Submitter: Add yourusecase cookbook to your assigned nodes

```
> knife node run_list add ACCEPTANCE-server 'recipe[yourusecase]'
> knife node run_list add UNION-server 'recipe[yourusecase]'
> knife node run_list add REHEARSAL-server 'recipe[yourusecase]'
> knife node run_list add DELIVERED-server 'recipe[yourusecase]'
```

```
uc1acceptance:
  run_list:
    recipe[bjc_bass]
    recipe[yourusecase]
```

GL: Submitter: List all of the Chef Automate environments

```
> knife environment list
```

```
_default
acceptance-automate-demo-automate-bjc-ecommerce-master
acceptance-automate-demo-automate-yourusecase-master
delivered
rehearsal
union
```

GL: Submitter: Add yourusecase cookbook to your assigned environments

```
> knife node environment_set ACCEPTANCE-server acceptance-automate-demo-automate-yourusecase-master
```

```
ACCEPTANCE-server:
  chef_environment: acceptance-automate-demo-automate-yourusecase-master
```

GL: Submitter: Add yourusecase cookbook to your assigned environments

```
> knife node environment_set UNION-server union
> knife node environment_set REHEARSAL-server rehearsal
> knife node environment_set DELIVERED-server union
```

```
REHEARSAL-server :
  chef_environment: rehearsal

DELIVERED-server:
  chef_environment: delivered
```

GL: Submitter: Kick off chef-client run on all systems

```
> knife job start 'chef-client' --search 'recipes:*'
```

```
WARNING: Falling back to Push Jobs v1 mode.
Started.  Job ID: 4e5675d348b35c00ecf9fc2f46f7c266
.Running (1/8 in progress) ...
.....................Complete.
```

EXERCISE

Group Lab: Iterating your code and submitting it through the Automate Workflow pipeline

Objective:

- ❑ Create your short lived feature branch
- ❑ Setup Kitchen
- ❑ Iterate the code in your project and test using Kitchen
- ❑ Submit your code into the Automate Workflow pipeline

CHEF

CONCEPT

Development Workflow

CHEF

Overview of the process going forward

CHEF

GL: Submitter: Go into your cookbook repo

```
C:\Users\chef\cookbooks> cd yourusecase
C:\Users\chef\cookbooks\yourusecase [master ≡]> ls
```

```
Directory: C:\Users\chef\cookbooks\yourusecase

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         12/9/2016  12:49 AM            .delivery
d----         12/9/2016   4:27 AM            .kitchen
d----         12/9/2016   6:00 PM            recipes
d----         12/8/2016  11:59 PM            spec
d----         12/8/2016  11:59 PM            test
-a---         12/8/2016  11:59 PM        142 .gitignore
-a---         12/9/2016   4:49 AM        781 .kitchen.yml
-a---         12/8/2016  11:59 PM         50 Berksfile
-a---         12/8/2016  11:59 PM       1240 chefignore
```

CHEF

GL: Submitter: Create a git Branch and Name it `FirstCommit`

```
~\cookbooks\yourusecase [master]> git checkout -b "FirstCommit"
```

```
Switched to a new branch 'FirstCommit'
```

CHEF

A Note About Test Kitchen

As you may know, whenever you modify a cookbook, you should test it locally with Test Kitchen.

The same holds true when using Chef Automate. You should test your cookbooks before submitting them to the pipeline.

CHEF

Submitter: Copy .kitchen_linux.yml config template

```
> cp ~/Desktop/Test_Kitchen/kitchen_linux.yml ~/cookbooks/yourusecase/.kitchen.yml
```


Submitter: Copy .kitchen_windows.yml config template

```
> cp ~/Desktop/Test_Kitchen/kitchen_windows.yml ~/cookbooks/yourusecase/.kitchen.yml
```


Submitter: Edit the Kitchen Configuration File

~/cookbooks/yourusecase/.kitchen.yml

```
...
platforms:
- name: ubuntu-14.04
    transport:
      username: ubuntu
      ssh_key: C:\Users\chef\.ssh\id_rsa
  - name: centos-6
    transport:
      username: root
      ssh_key:  C:\Users\chef\.ssh\id_rsa
    driver_config:
      user_data: C:\Users\chef\user_data...
```

https://www.centos.org

CHEF

Submitter: Edit the Kitchen Configuration File

~/cookbooks/yourusecase/.kitchen.yml

```
...
platforms:
- name: centos-7
    transport:
      username: root
      ssh_key:  C:\Users\chef\.ssh\id_rsa
    driver_config:
      user_data: C:\Users\chef\user_data

...
```

https://www.centos.org

CHEF

Submitter: Edit the Kitchen Configuration File

~/cookbooks/yourusecase/.kitchen.yml

```
...
suites:
  - name: default
    run_list:
      - recipe[yourusecase::default]
    verifier:
      inspec_tests:
        - test/recipes
    attributes:
```


CHEF

Submitter: Confirm your Kitchen config is correct

```
> kitchen list
```

```
Instance          Driver   Provisioner  Verifier  Transport  Last Action
default-centos-7  Ec2      ChefZero     Inspec    Ssh        <Not Created>
```

CHEF

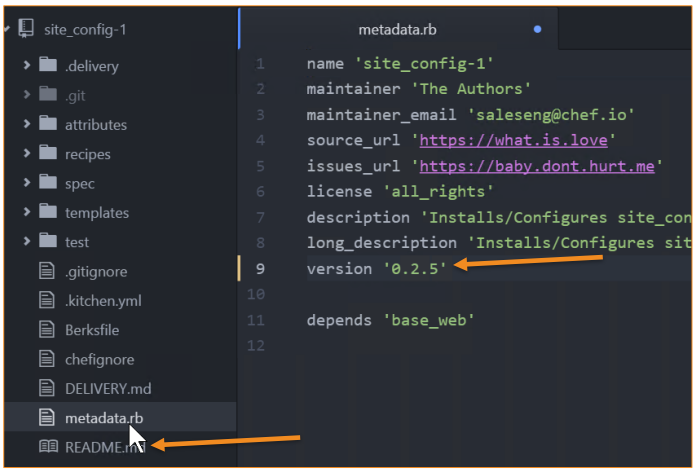
GL: Submitter: Run VS Code to write your code

```
C:\Users\chef\cookbooks\yourusecase [master ≡]> code .
```

GL: Submitter: Always bump the metadata.rb

Edit the **metadata.rb** and update the version (e.g., 0.1.1)

Take a moment to update the **README.md** as well.

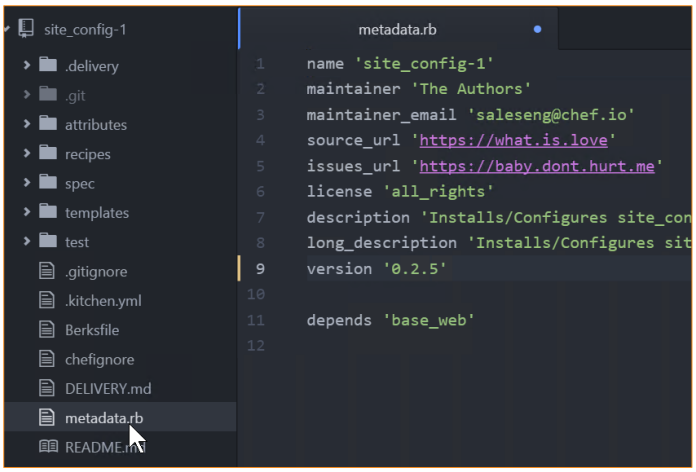


Remember to save your changes in VS Code.

For example, **Ctrl + S** or

GL: Submitter: Work on your code!

Make with the typey typey and iterate your code!



Submitter: Converge the test instance

```
> kitchen converge
```

```
...
-----> Starting Kitchen (v1.13.2)
-----> Converging <default-centos-6>...
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 5.1.0...
       Removing non-cookbook files before transfer
       Preparing validation.pem
       Preparing client.rb
-----> Chef Omnibus installation detected (install only if missing)
```

Submitter: Login to your test instance for a look

```
> kitchen login
```

```
Last login: Fri Dec  9 18:27:57 2016 from ec2-54-202-204-187.us-west-
2.compute.amazonaws.com
[centos@ip-172-31-45-104 ~]$
```

OPTION: Retrieve the FQDN for your instance

```
> cat .kitchen/default-centos-7.yml
```

```
---
username: centos
server_id: i-d63ba942
hostname: ec2-54-202-252-202.us-west-2.compute.amazonaws.com
last_action: converge
```

GL: Submitter: Git commit with your note

```
...[yourusecase [FirstCommit +0 ~2 -0 !]> git add .
...[yourusecase [FirstCommit +0 ~2 -0 !]> git commit -m "Initial commit"
```

```
[FirstCommit 1b435bb] Initial commit
 2 files changed, 2 insertions(+), 2 deletions(-)
```

GL: A Note About 'delivery review'

In a moment you will run the `delivery review` command.

The `delivery review` command submits the change to Delivery and kicks off the pipeline.

This command is the equivalent to `git push`, although it also creates a change in Chef Automate that is similar to a pull request in GitHub and other git-based version control systems.

GL: Submitter: Run `delivery review`

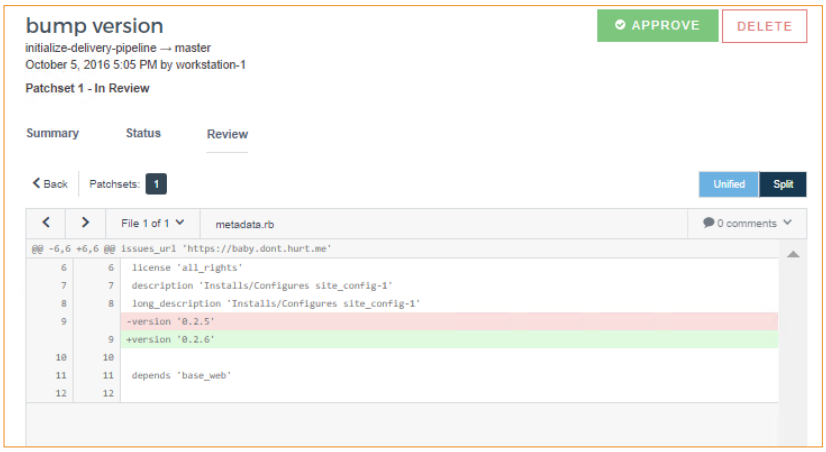
```
...\cookbooks\yourusecase [FirstCommit]> delivery review
```

```
Chef Delivery
Loading configuration from C:\Users\chef\cookbooks\yourusecase
Review for change FirstCommit targeted for pipeline master
Created new patchset
https://automate.automate-demo.com/e/automate-demo/#/organizations/automate/projects/yourusecase/changes/eb770837-b31c-4c4d-8c3c-47d14eeee981
```

CHEF

GL: Reviewer: Review the Changes

Now you can see the changes that are pending your approval.

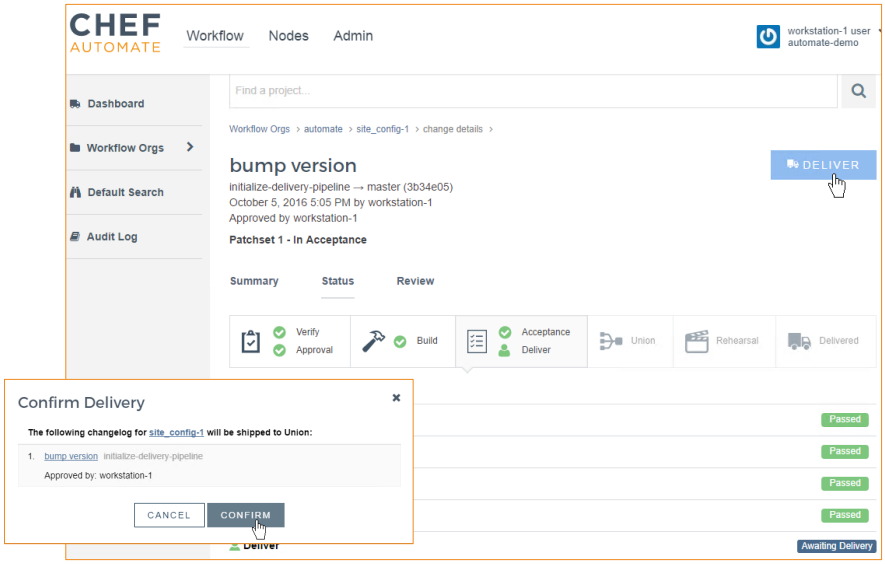


CHEF

GL: Reviewer: Deliver the Artifact

Reviewer: Click the **Deliver** button and the resulting **Confirm** button.

This action confirms that the artifact can be released. The artifact will move through the next three pipeline stages automatically.



GL: Submitter: Integrate the Change to Your Local Master Branch

```
> git checkout master
```

```
Switched to branch 'master'
Your branch is up-to-date with 'delivery/master'.
```

GL: Submitter: Integrate the Change to Your Local Master Branch

```
> git pull --prune
```

```
From ssh://automate-demo@automate.automate-demo.com:8989/automate-demo/automate/site_config-1
 x [deleted]         (none)     -> delivery/_for/master/initialize-delivery-pipeline
remote: Counting objects: 2, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
   d01bece..3b34e05  master     -> delivery/master
Updating d01bece..3b34e05
Fast-forward
 metadata.rb | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Submitter: Lather Rinse Repeat cheatsheet

> cd ~/cookbooks/yourusecase
> git checkout -b “YourFancyBranchName”
> code .
> kitchen converge
> git add .
> git status
> git commit -m "Release version 0.2.0”
> delivery review

[Peer Review/Deliver in Automate Workflow UI]

> git checkout master
> git pull delivery master --prune

Q&A

What questions can we answer for you?

- Create the project and the pipeline.
- `delivery init` command.
- `delivery review` command.
- Approve the change.
- Deliver the change.
- Integrate the change to your local master branch.

CHEF

CHEF™

Further Resources

Other Places to Talk About, Practice, and Learn Chef

LAB

Going Forward

There are many Chef resources available to you outside this class. During this module we will talk about just a few of those resources.

But…remember what we said at the beginning of this class:

The best way to learn Chef is to use Chef

Practice Chef

First, let's talk about stuff you can read to help you learn Chef.

EXERCISE

learn.chef.io

Interactive learning for those new to Chef.

EXERCISE

Beyond Essentials

What happens during a knife bootstrap?
What happens during a chef-client run?
What is the security model used by chef-client
Explains Attribute Precedence

https://learn.chef.io/skills

 359

EXERCISE

Community Resources

Example Cookbooks, Articles, Podcasts, and More

https://github.com/obazoud/awesome-chef

 360

Resources You Can Read

A lot of people in the Chef community have written about Chef.

Here are just a few of those resources.

DOCS

docs.chef.io

Docs are available to you, 24 hours a day, 7 days a week.

Any question you have, you probably will find the answer for on our Docs site.

DOCS

inspec.io

Tutorials and documentation on how to use Inspec to test your infrastructure and build comprehensive compliance profiles.

CHEF

Chef Certification Levels

A defined Certification Level is awarded after obtaining a set number of badges.

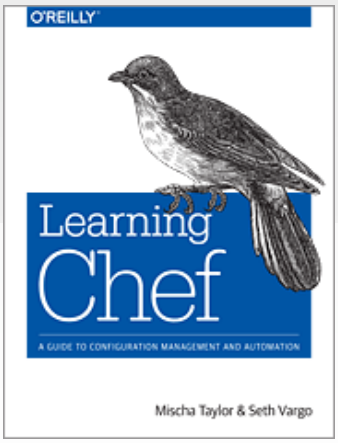
CHEF

Who doesn't love T-shirts and Stickers

CHEF

Learning Chef

A Guide to Configuration Management and Automation

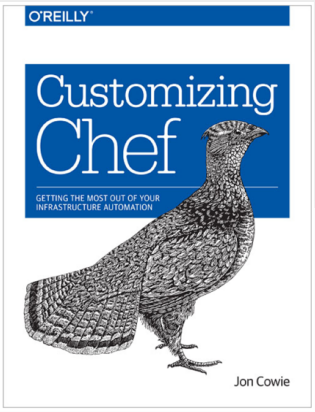


6366

CHEF

Customizing Chef

Getting the Most Out of Your Infrastructure Automation

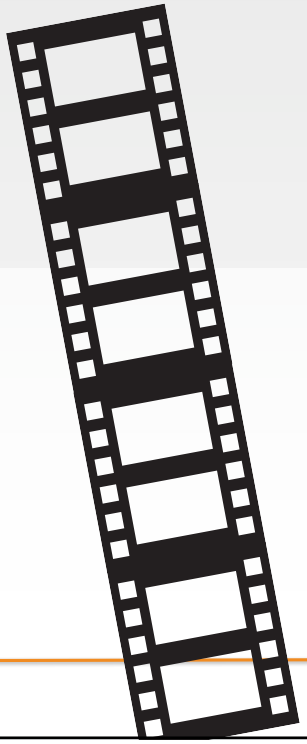


CHEF

YouTube Channel

- ChefConf Talks
- Training Videos

https://www.youtube.com/user/getchef/playlists



CHEF

foodfightshow.org

The Podcast where DevOps chefs do battle

Chef Developers' IRC Meeting

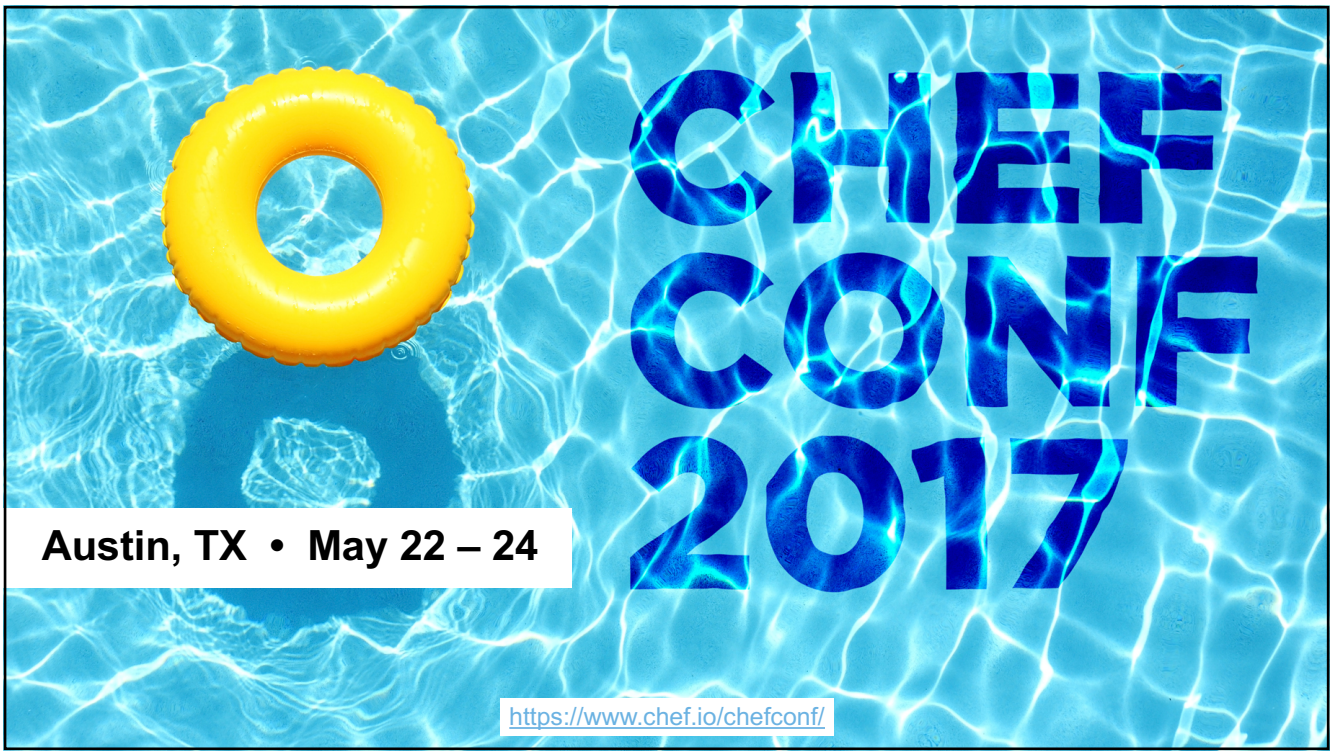
https://github.com/chef/chef-community-irc-meetings

Chef Product Feedback Forum

Create your product feature ideas for the Chef engineering teams. As a registered user, you'll be able to vote on your features and the features proposed by others...

https://feedback.chef.io

CHEF




CHEF™
Thank You!
©2016 Chef Software Inc.
6-
CHEF