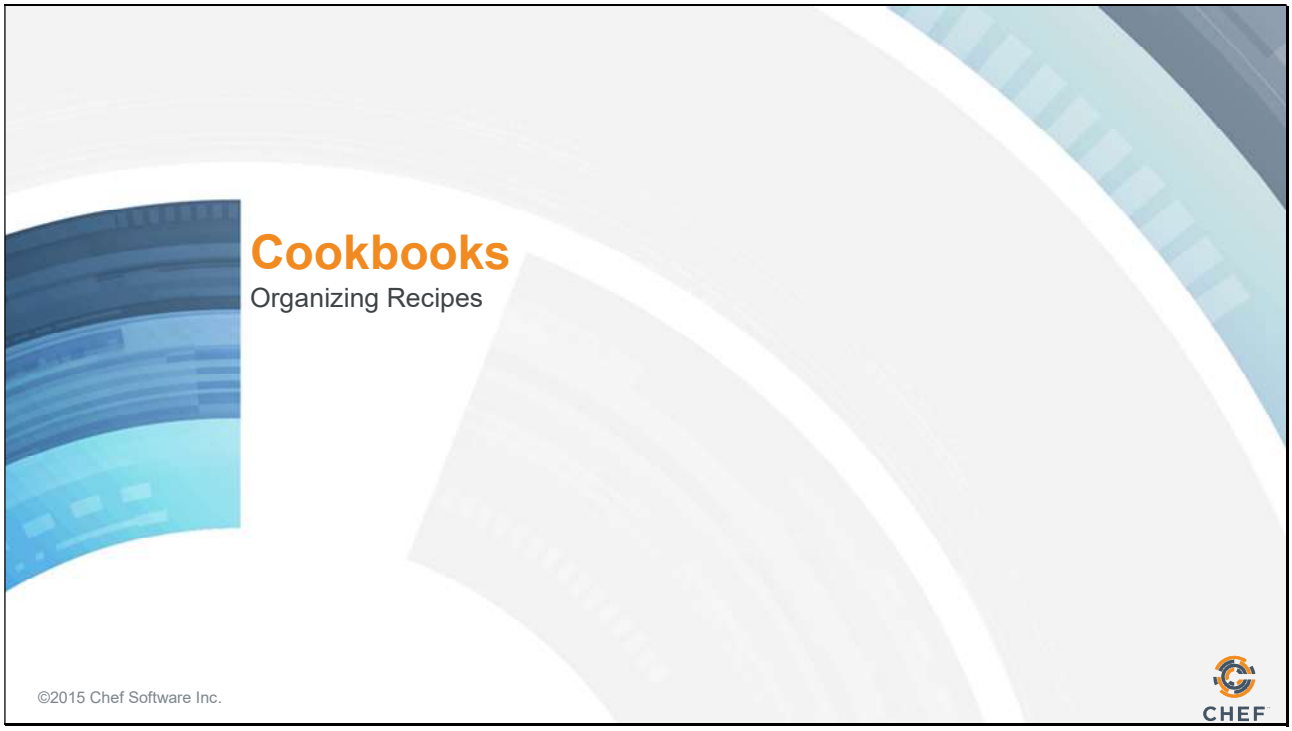
3: Cookbooks



Slide 2

Objectives

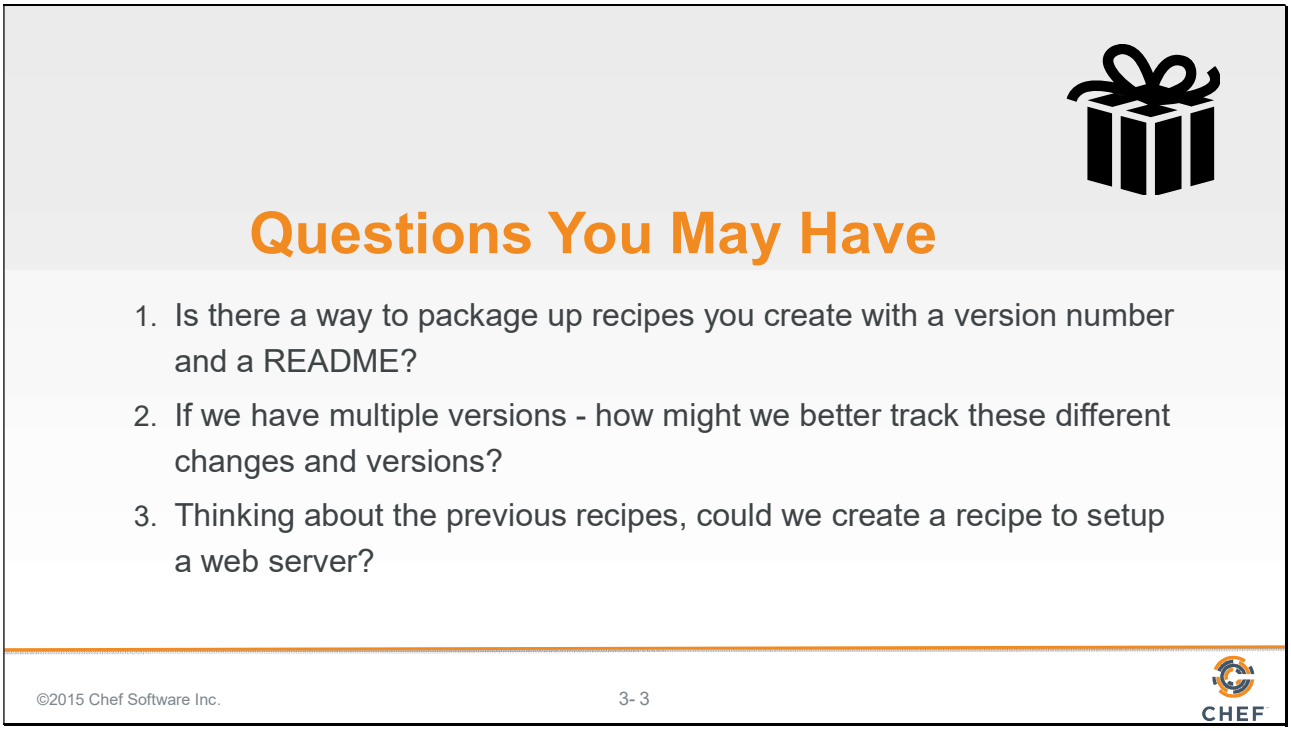
After completing this module, you should be able to:

- Use version control
- Generate a Chef cookbook
- Define a Chef recipe that sets up a web server

CHEF

In this module you will learn how to use version control, generate a Chef cookbook and define a Chef recipe that sets up a web server.

Slide 3

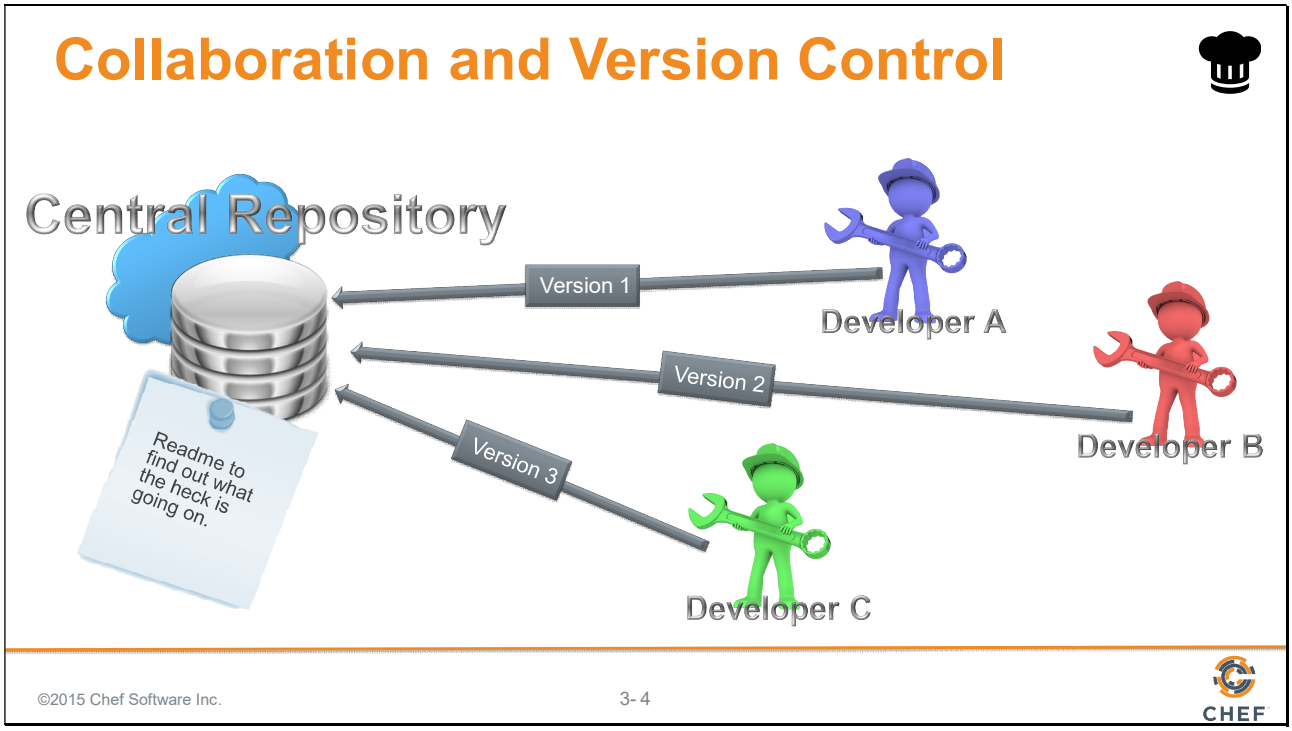


1. Recipes on their own would be difficult to manage. Imagine handing a copy of a recipe that you develop to an individual. A few weeks later they come back to you and ask you why it no longer works. If you had made a number of changes since the last time you talked with them how do you know which version you gave them? Adding a version number and a README would allow you to better document the features of a particular recipe.

2. When developing recipes it is important to consider using a version control solution that allows us to track the changes that we make.

3. The recipe that you put together to disable UAC showed that Chef is powerful enough to manage the registry. Chef is powerful enough to install and configure a web server.

Slide 4



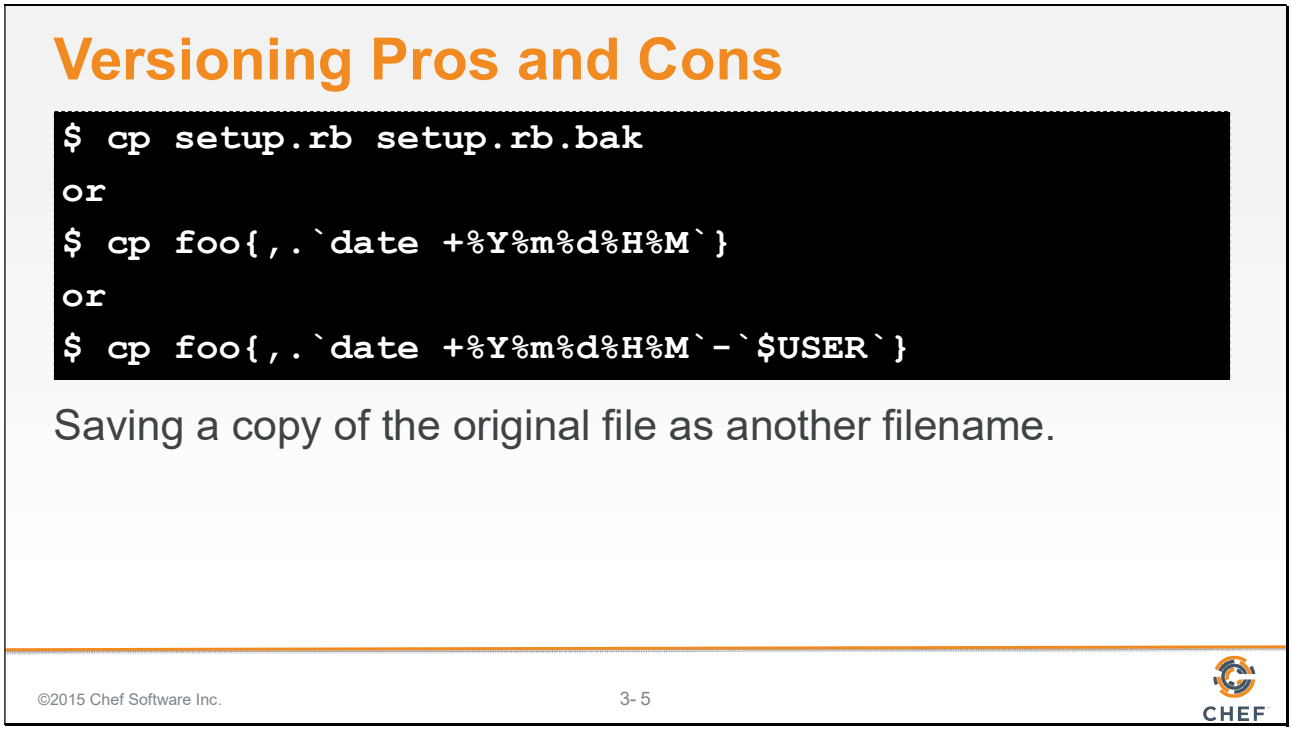
Before we answer that question, let's talk about collaboration. Usually, none of us work in a vacuum, and it's important that systems are in place to make collaboration easier. One such system is versioning. Versioning will make it easier to share the recipes that we create.

A versioning system should include:

A Central Repository into which all the developers publish their work.

Each revision should be stored as a new version.

For each change, a commit message should be added so that everyone knows what has or has not been changed



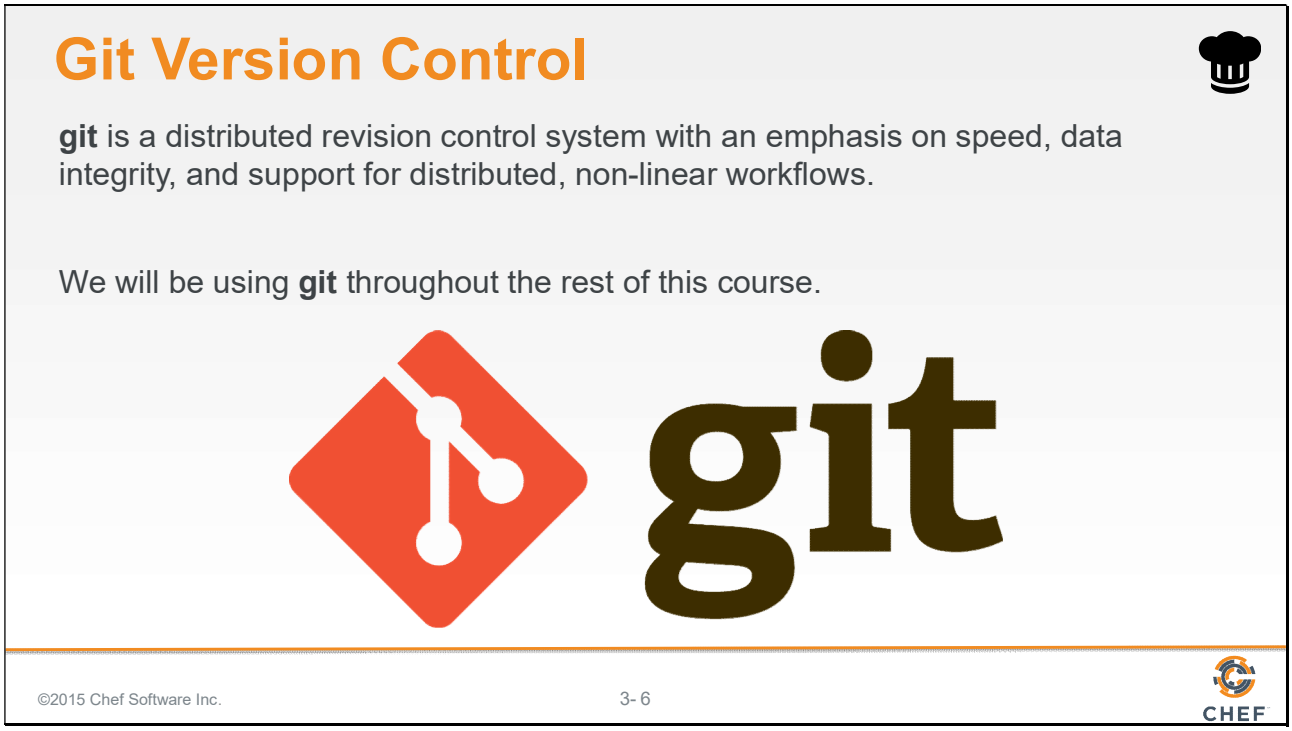
Let’s explore this first option of renaming the file by adding a quick extension, like in the first example shown here. In this way we can keep working on the original file as we add more features. As a group let’s talk about the pros and cons of using this strategy.

So obviously a single backup won't do. We need backups more often as we are going to be iterating quickly.

We could use the current date and time down to the minute like in the second example. As a group let’s talk about the pros and cons of using this strategy.

Would adding the user's name to the end of the file, like in the third example, solve the problems we are facing with other choices? Again what are the pros and cons of this new approach?

Slide 6

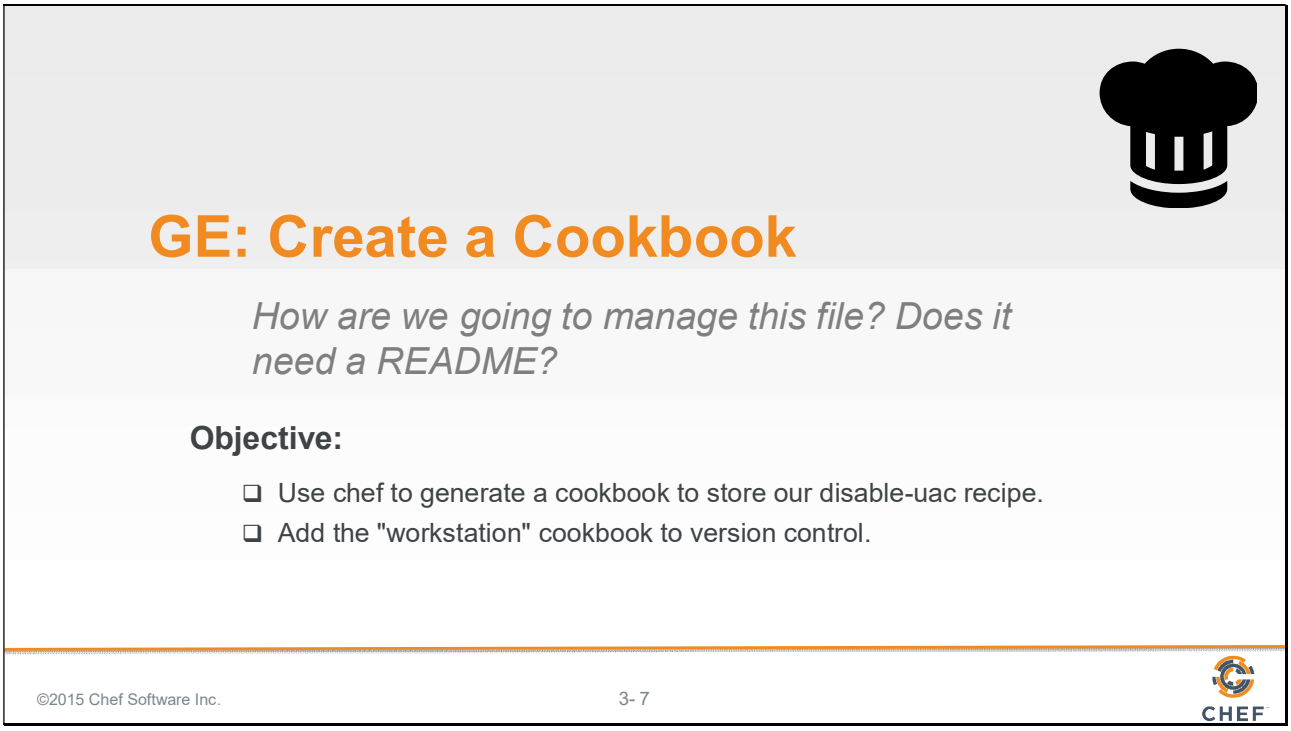


How about we use git?

What are the pros and cons of this approach?

For the rest of this course we will be using git. This may not be the version control software you use on your teams or within your organization and that is alright. Our use of git within this course is used solely to demonstrate the use of version control when developing Chef code. When you develop with Chef you are welcome to use the version control system of your choice.

Slide 7

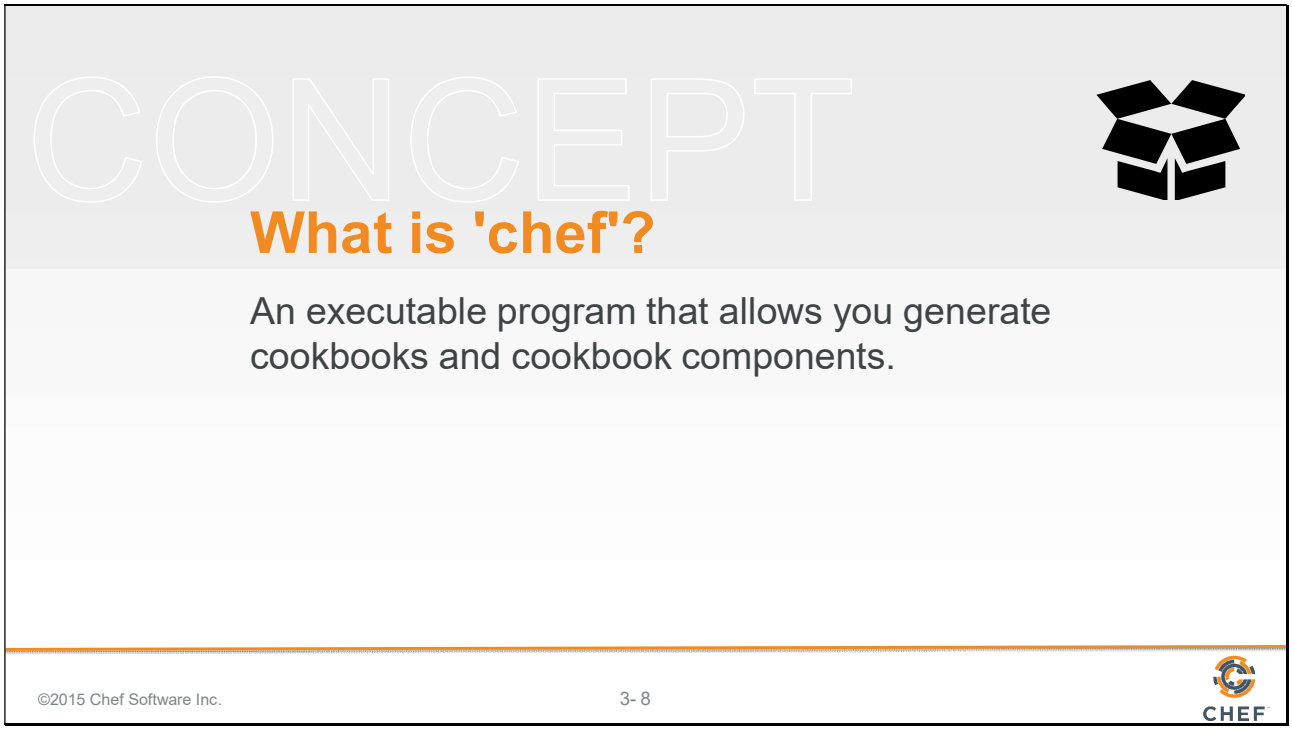


The setup recipe now installs everything we currently need on our workstation.

But before throw this recipe file into a directory with our other scripts we should look at a concept in Chef called a cookbook.

What is a cookbook? How do we create one? Let's ask 'chef'.

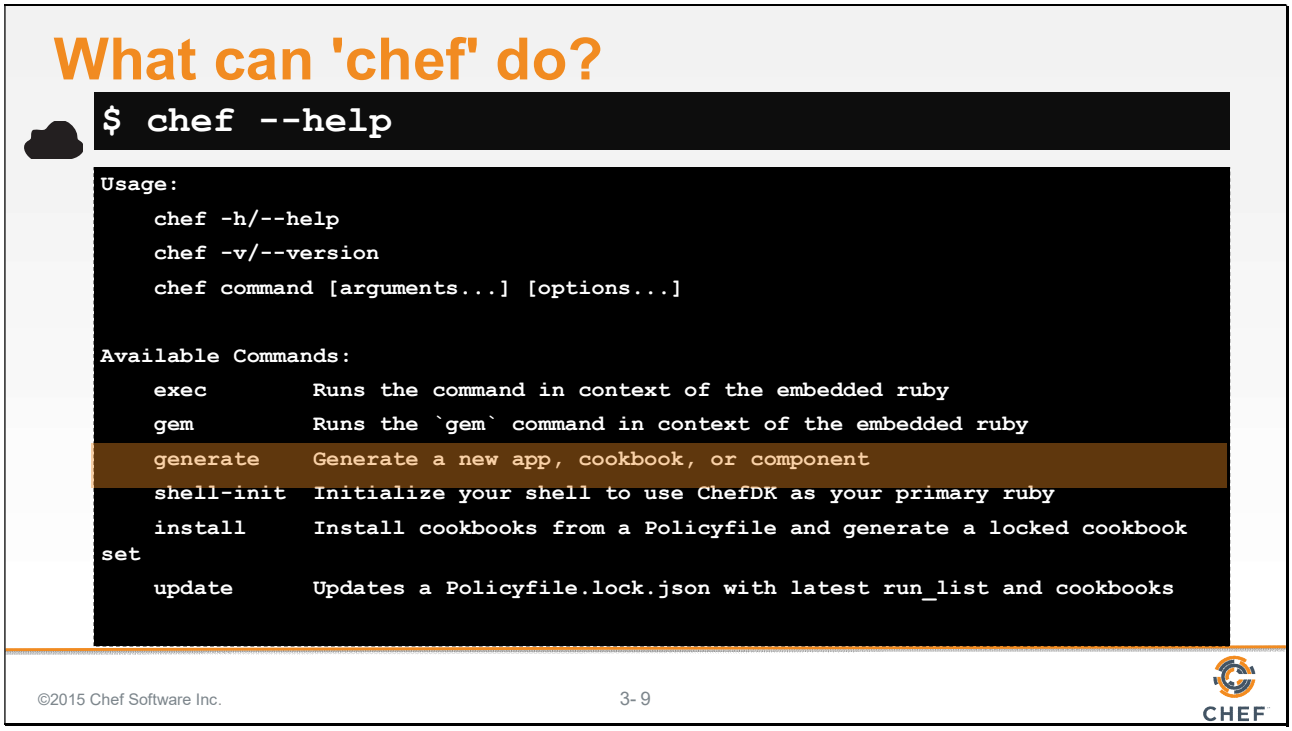
Slide 8



In this context, 'chef' is a command, not the company.

What's the best way to learn Chef? Use Chef. We want you to literally run 'chef'.

Slide 9



'chef' is a command-line application that does quite a few things. The most important thing to us right now is its ability to generate cookbooks and components.

Alright. So 'chef' can generate a cookbook. But what is the purpose of a cookbook? That sounds like we should read the documentation.

Slide 10

Cookbooks

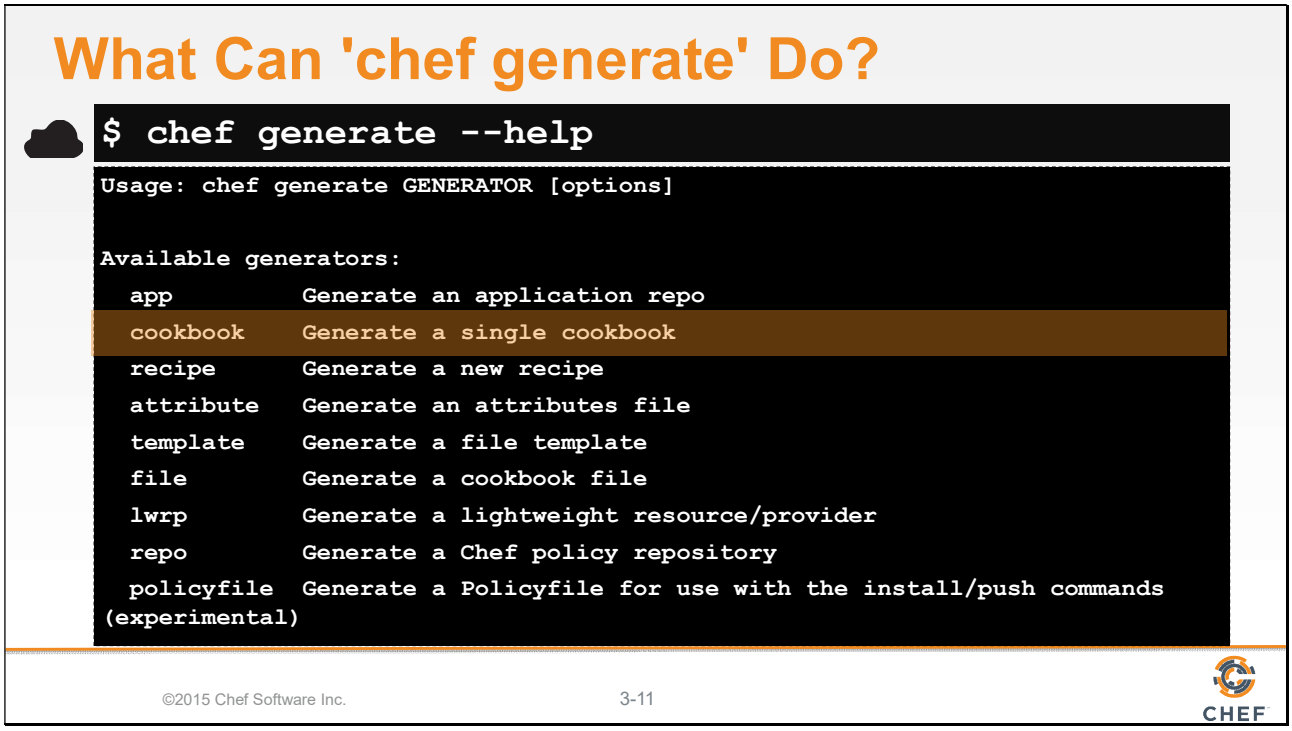
A Chef cookbook is the fundamental unit of configuration and policy distribution.

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.

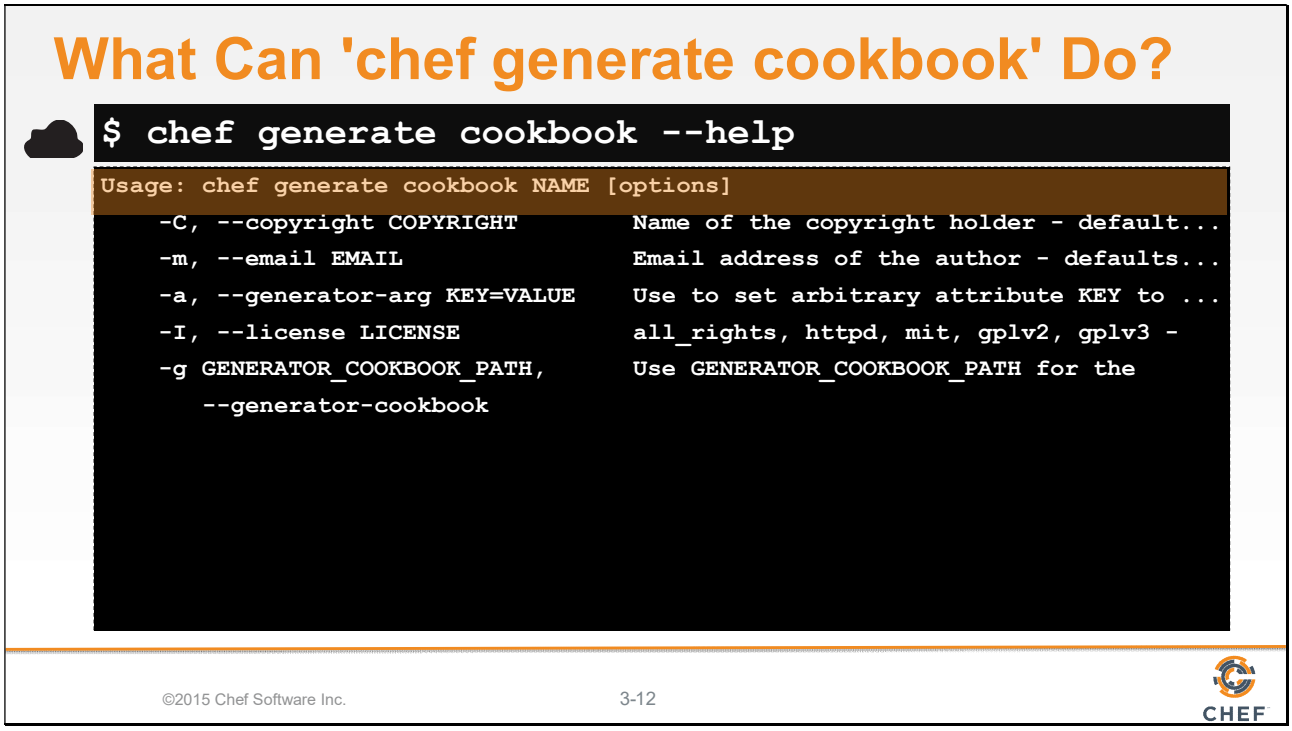
Read the first three paragraphs here: http://docs.chef.io/cookbooks.html

It's important that you learn to read the Chef documentation. Let's look up cookbooks in Chef's documentation. Visit the docs page on cookbooks and read the first three paragraphs.

A cookbook is a structure that contains recipes. It also contains a number of other things--but right now we are most interested in a finding a home for our recipes, giving them a version, and providing a README to help describe them.



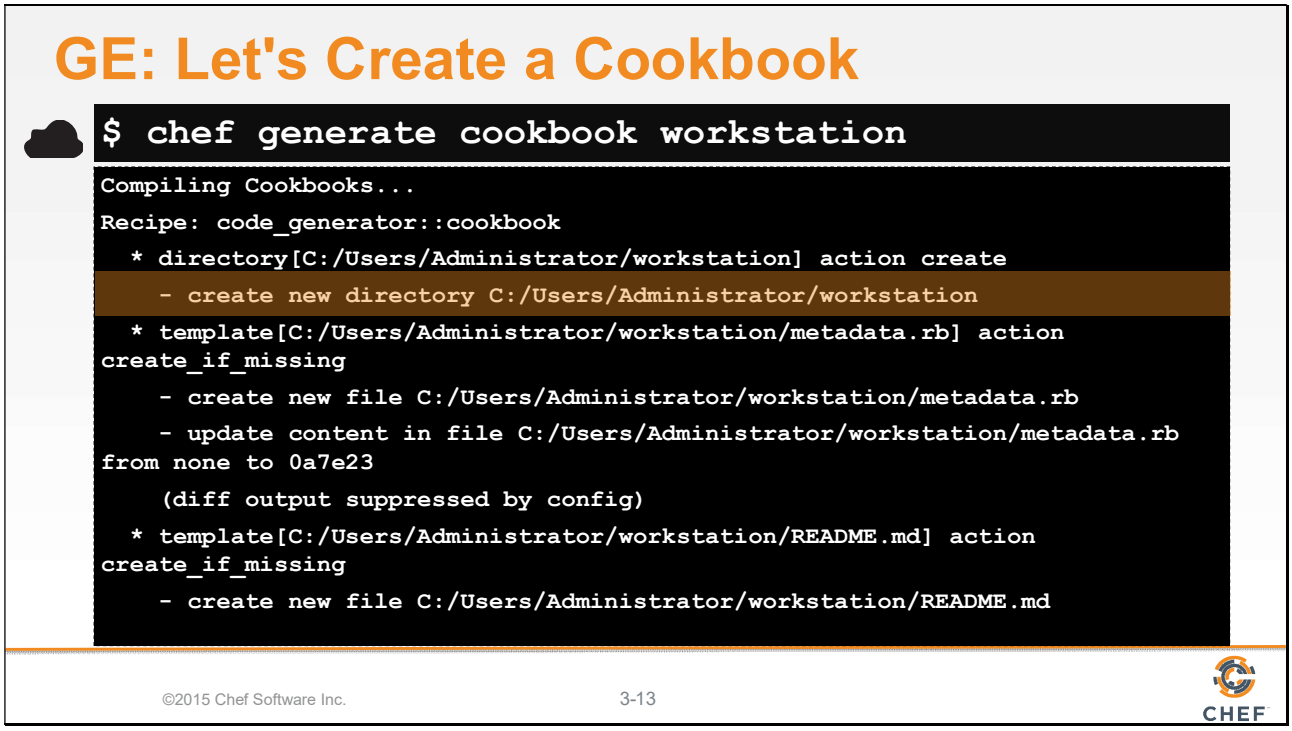
Let's examine the '`chef generate`' command. We can see that the command is capable of generating a large number of different things for us. It looks like if we want to generate a cookbook we're going to need to use 'chef generate cookbook'.



Let's ask the 'chef generate cookbook' command for help to see how it is used.

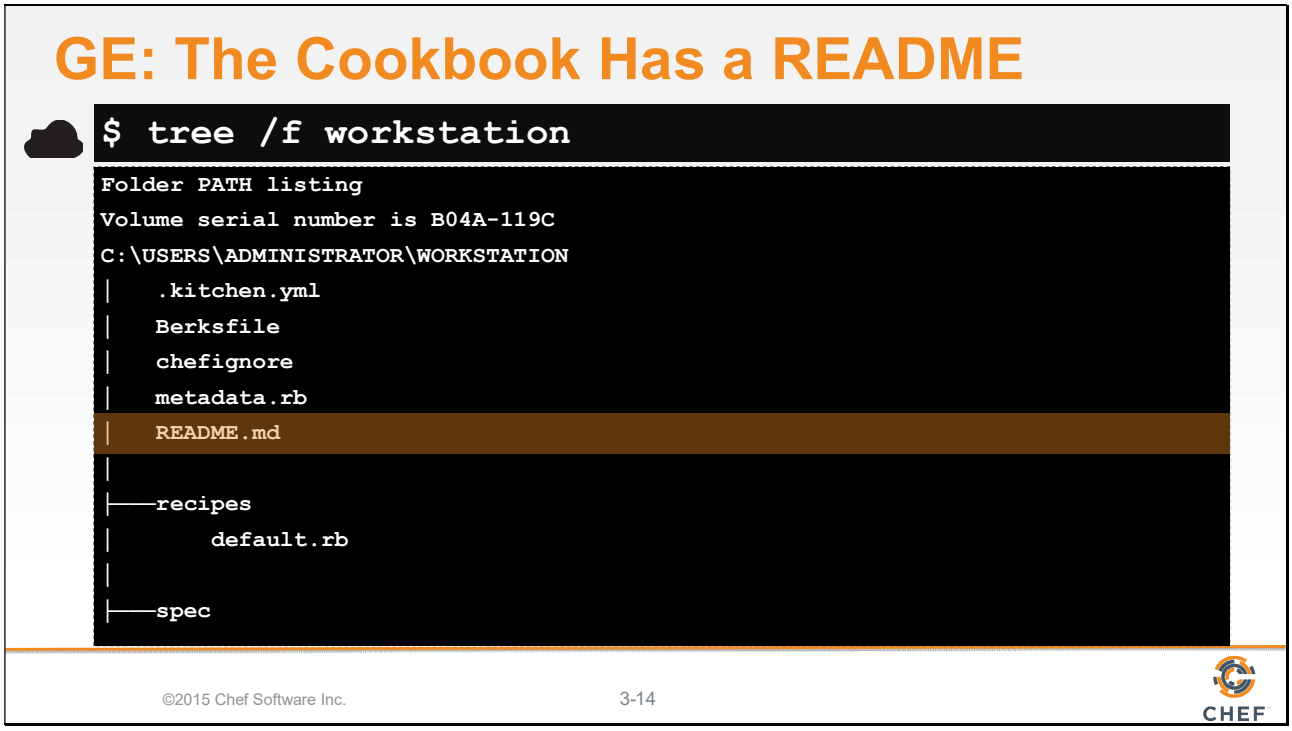
To generate a cookbook, all we have to do is provide it with a name.

There are two hard things in Computer Science and one of those is giving something a name.



We have you covered. Call the cookbook *workstation*. That's a generic enough name.

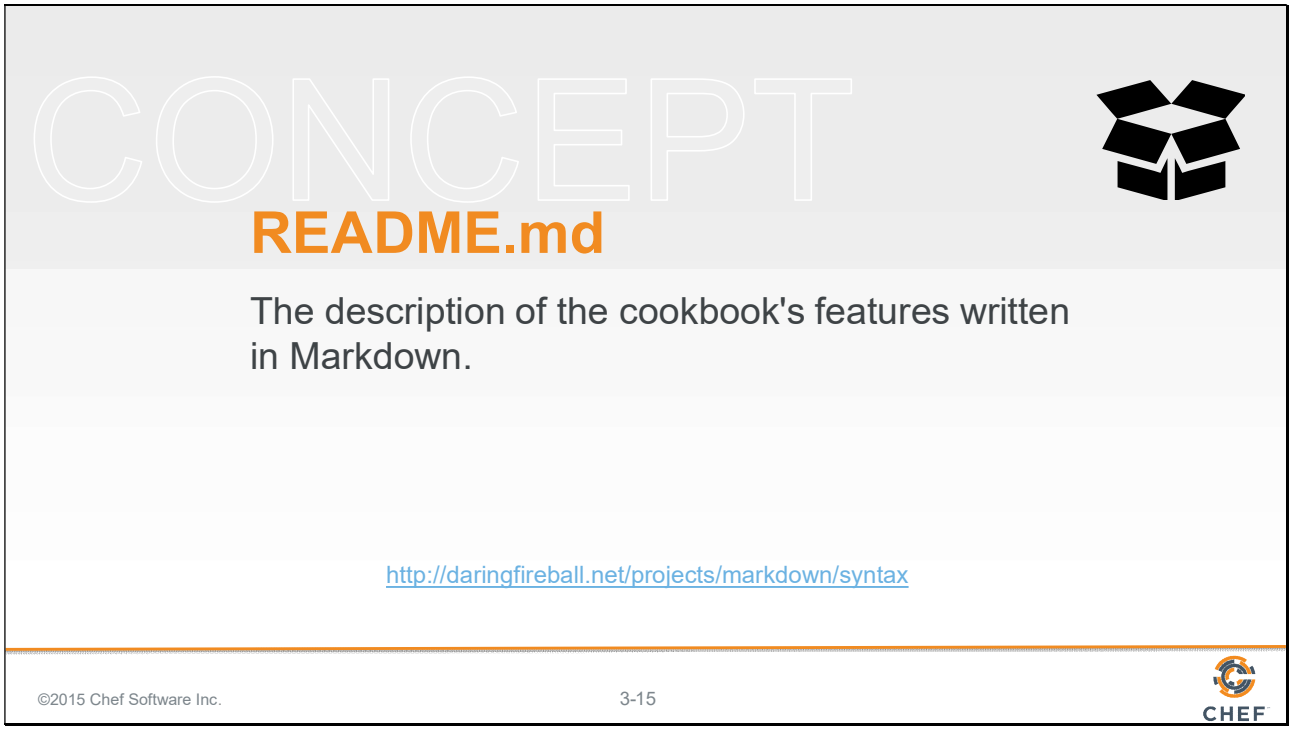
We want you to use '`chef generate`' to generate a cookbook named *workstation*.



Aren't you curious what's inside it? Let's take a look with the help of the '`tree`' command. If we provide '`tree`' with a path we will see all the visible files in the specified directory.

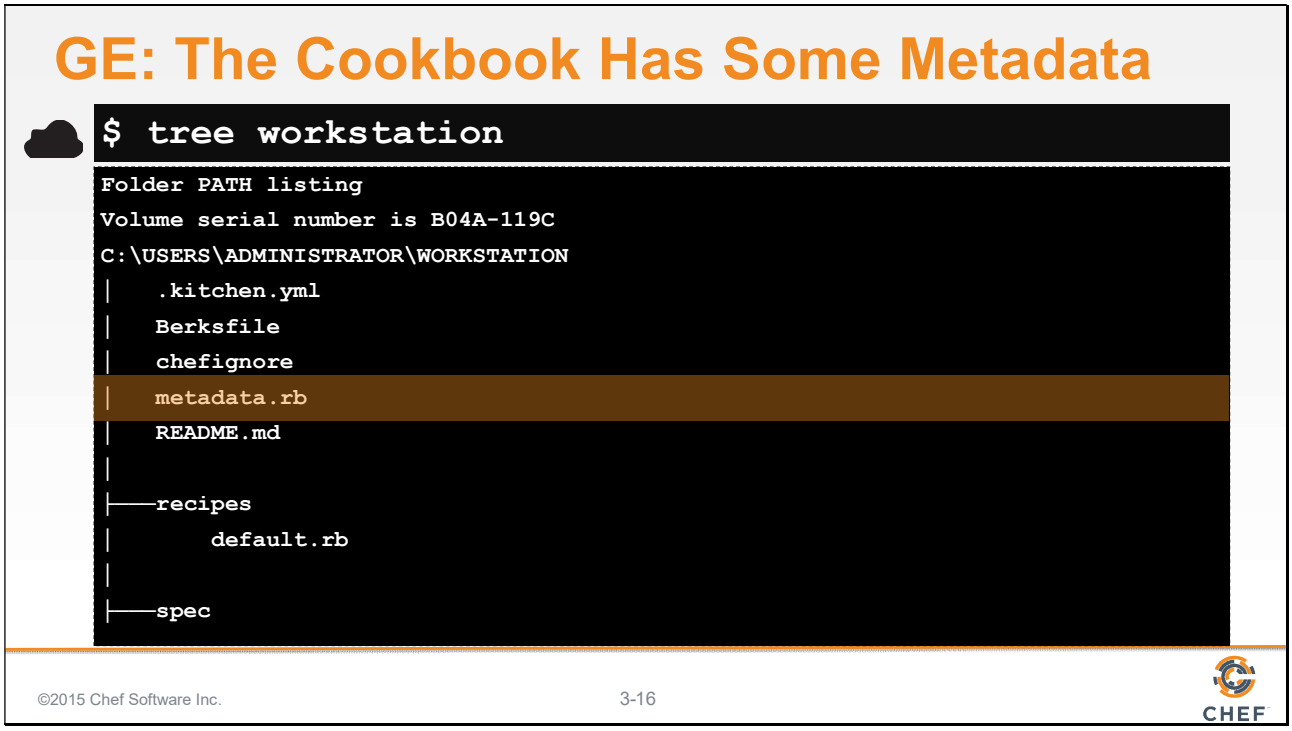
So the chef cookbook generator created an outline of a cookbook with a number of default files and folders. The first one we'll focus on is the README.

Slide 15



All cookbooks that 'chef' will generate for you will include a default README file. The extension `.md` means that the file is a markdown file.

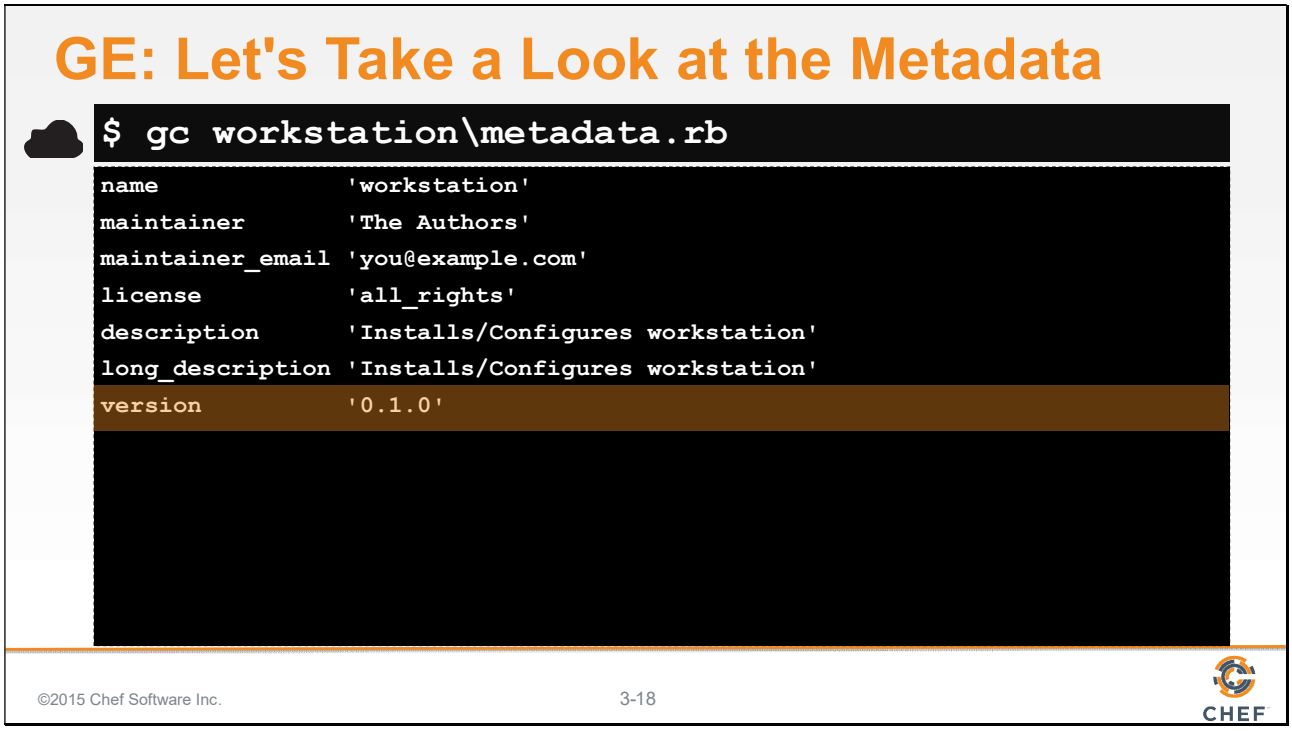
Markdown files are text documents that use various punctuation characters to provide formatting. They are meant to be easily readable by humans and can be easily be rendered as HTML or other formats by computers.



The cookbook also has a metadata file.

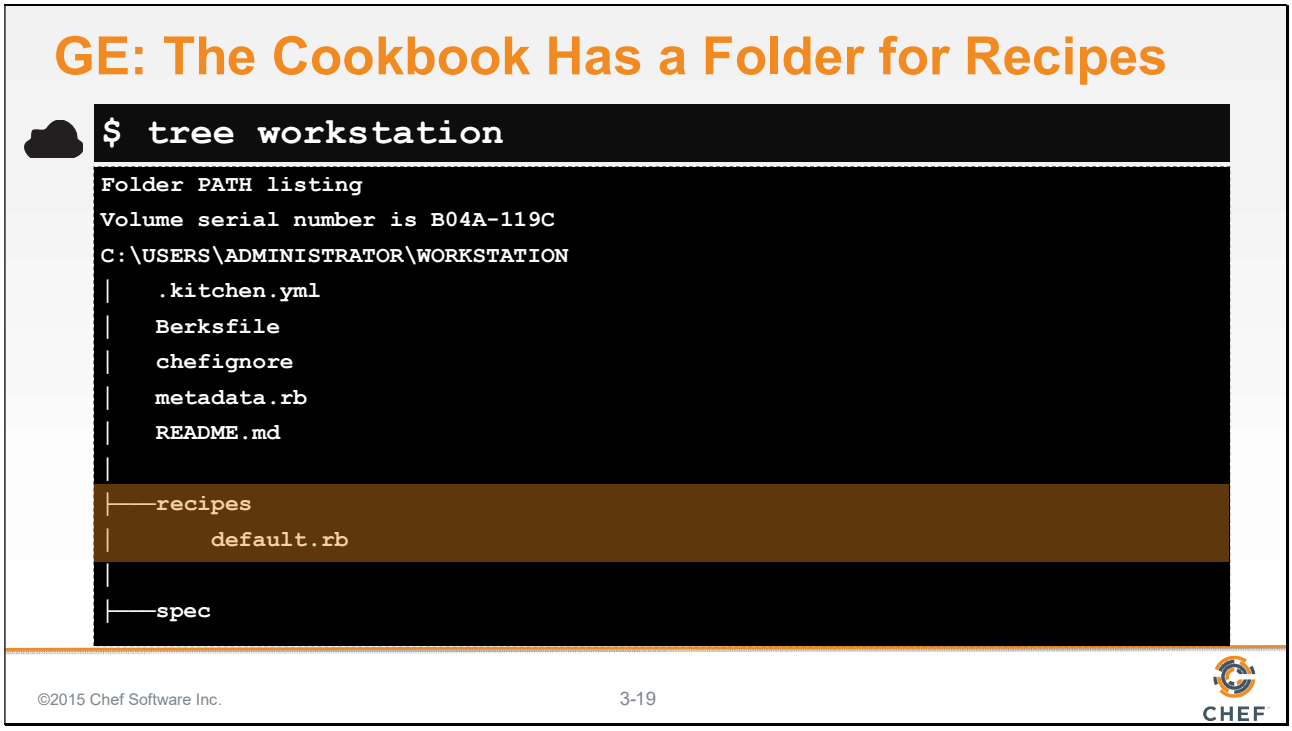
Slide 17

This is a ruby file that contains its own domain specific language (DSL) for describing the details about the cookbook.



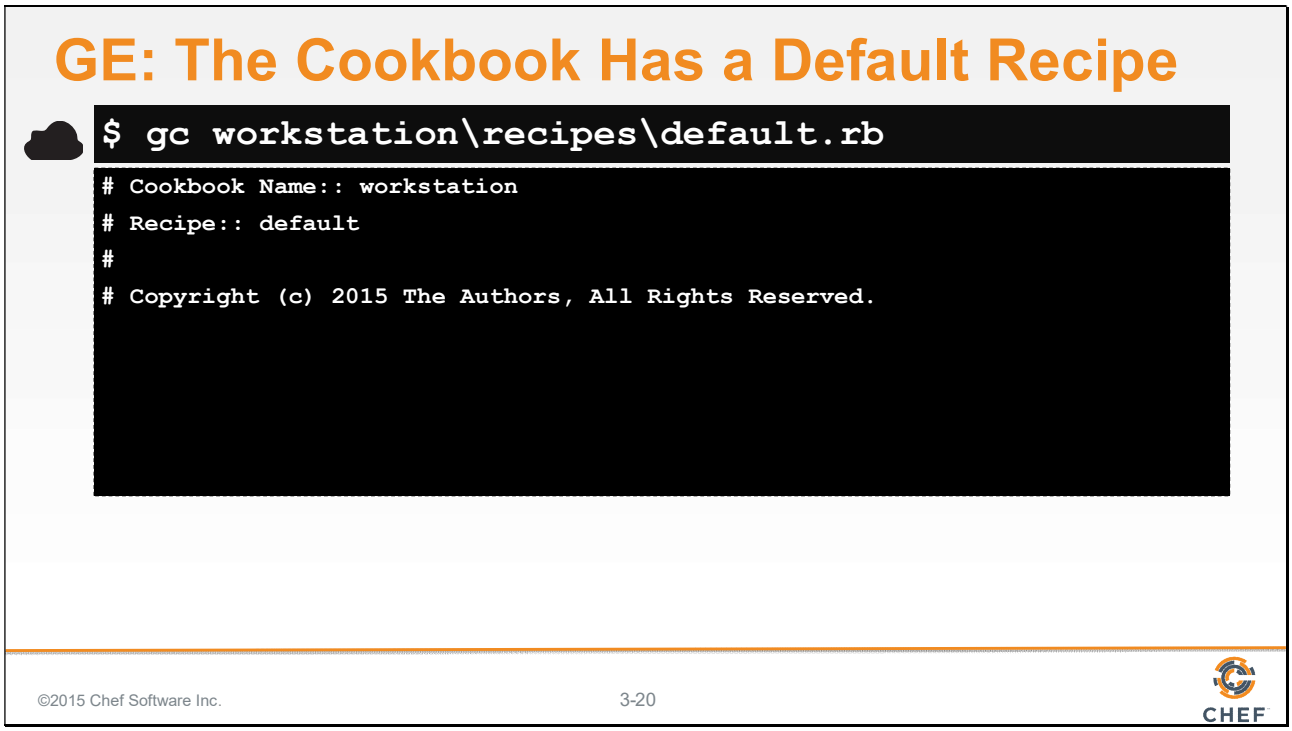
If you view the contents of your new cookbook's metadata, you'll see a number of details that help describe the cookbook:

The name of the cookbook, its maintainer, a way to reach them, how the cookbook is licensed, descriptions, and the cookbook's version number.



The cookbook also has a folder named *recipes*. This is where we store the recipes in our cookbook. You'll see that the generator created a default recipe in our cookbook. What does it do?

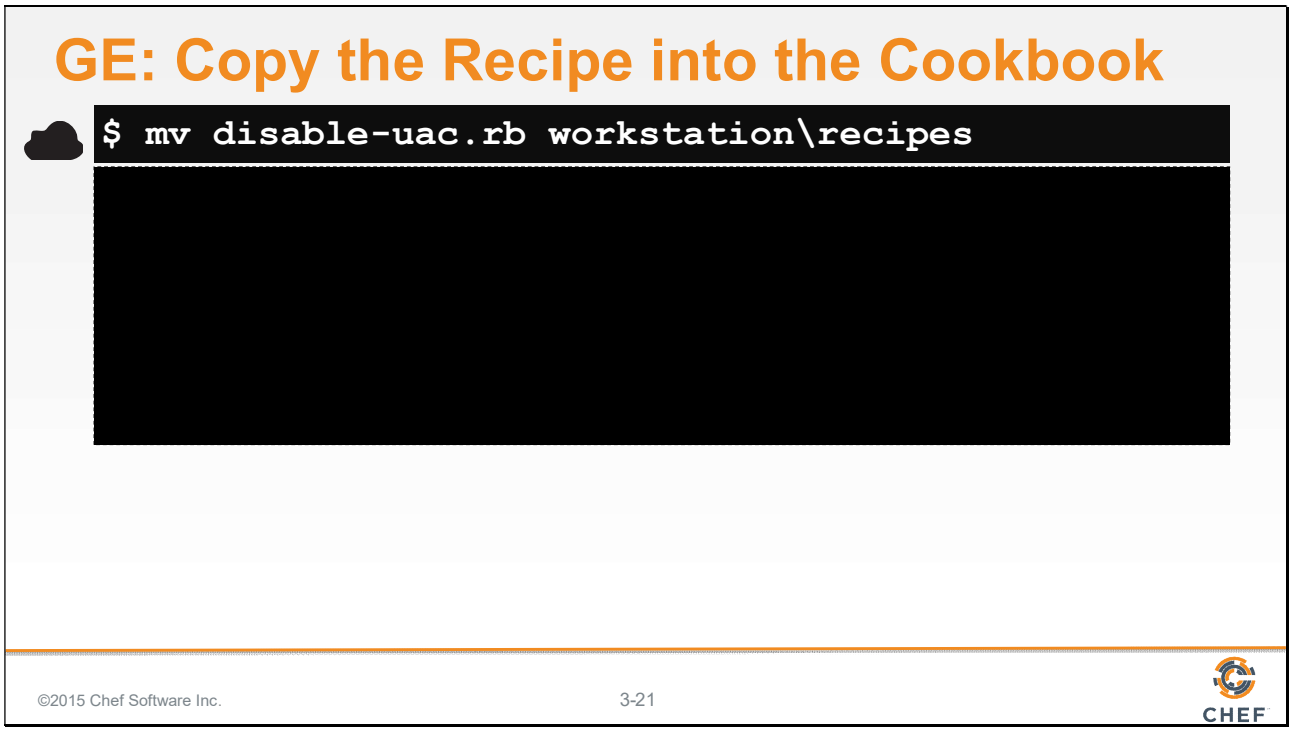
Slide 20



Looking at the contents of the default recipe you'll find it's empty except for some ruby comments.

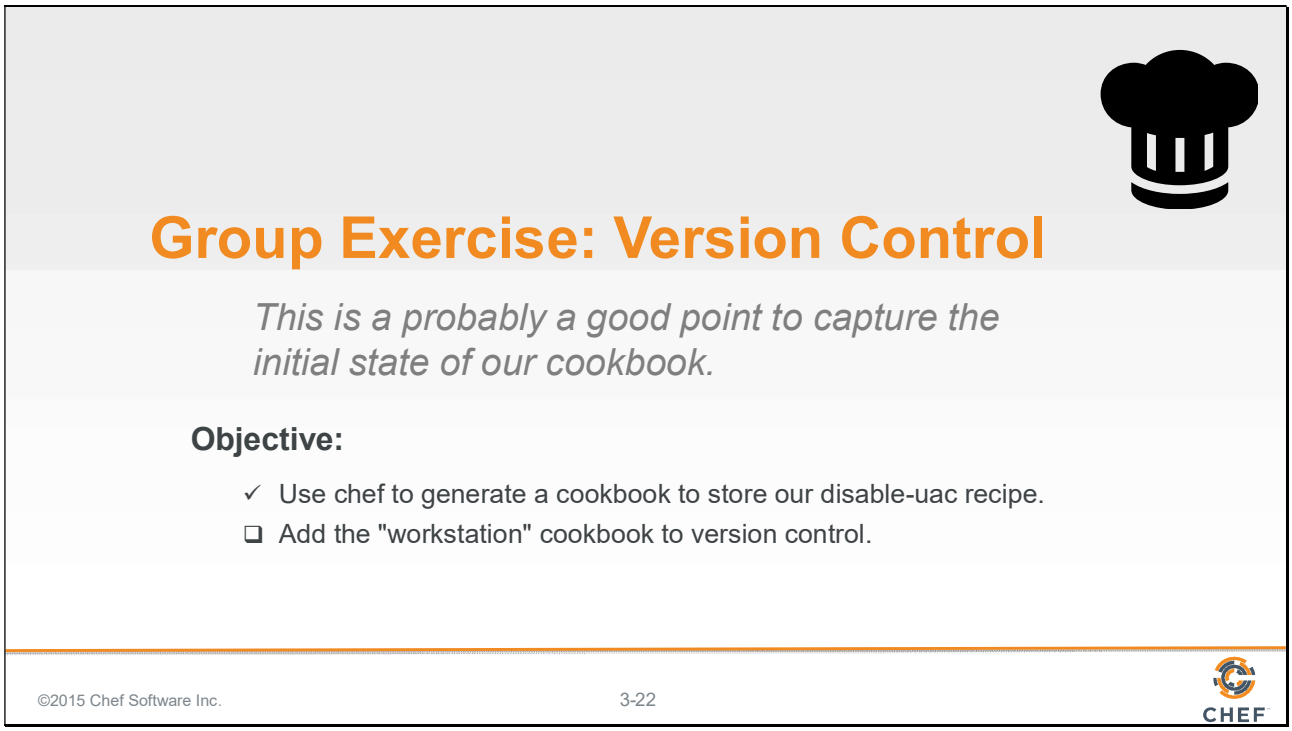
A cookbook doesn't have to have a default recipe but most every cookbook has one. It's called *default* because when you think of a cookbook, it is probably the recipe that defines the most common configuration policy.

Slide 21



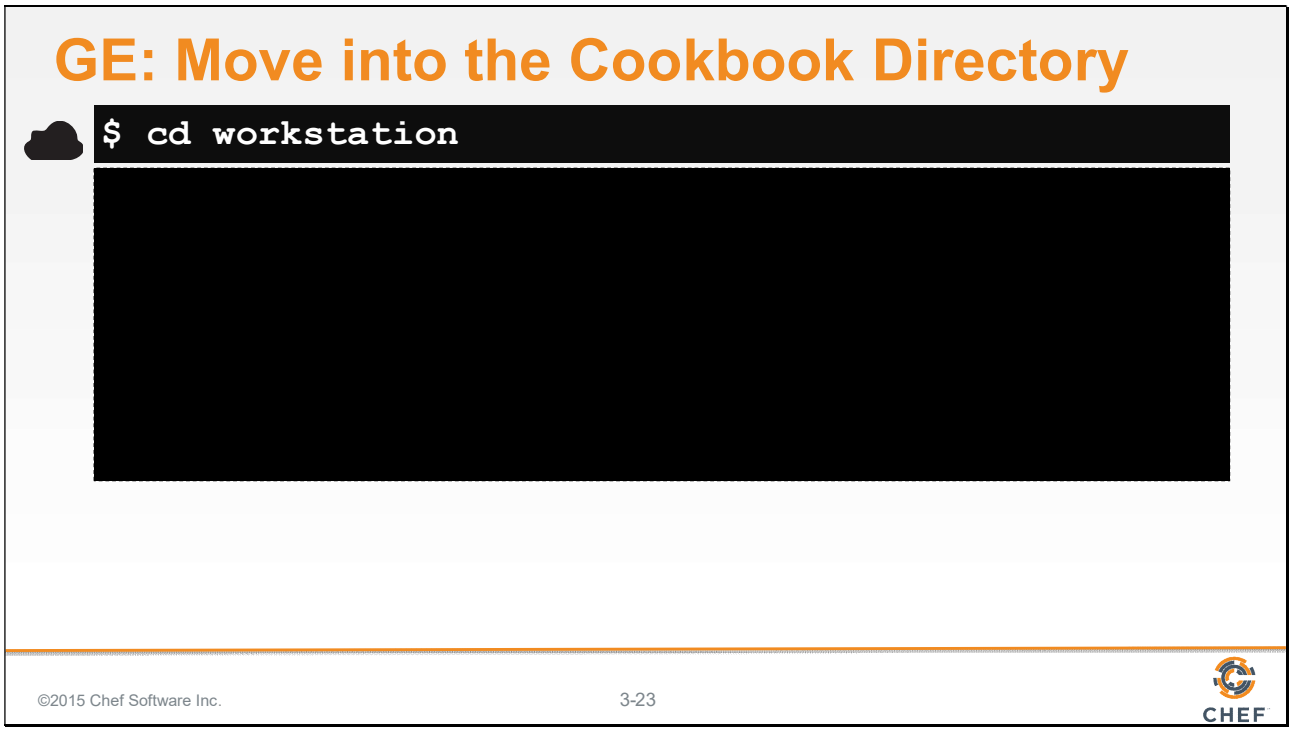
From the Home directory, move your 'disable-uac.rb' recipe to the workstation cookbook and place it alongside our default recipe.

Slide 22



Now that we have our cookbook with its README and version number, it's time to start tracking our changes with git.

Slide 23



Change into the workstation cookbook directory.

Slide 24

We want git to start tracking the entire contents of this folder and any content in the subfolders. To do that with git, you need to execute the command 'git init' in the parent directory of the cookbook that you want to start tracking.

You will notice that git will say that the repository has been 'Reinitialized'. This is because the chef cookbook generator detected that we have git installed and automatically initialized the cookbook as a git repository.

Slide 25

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.

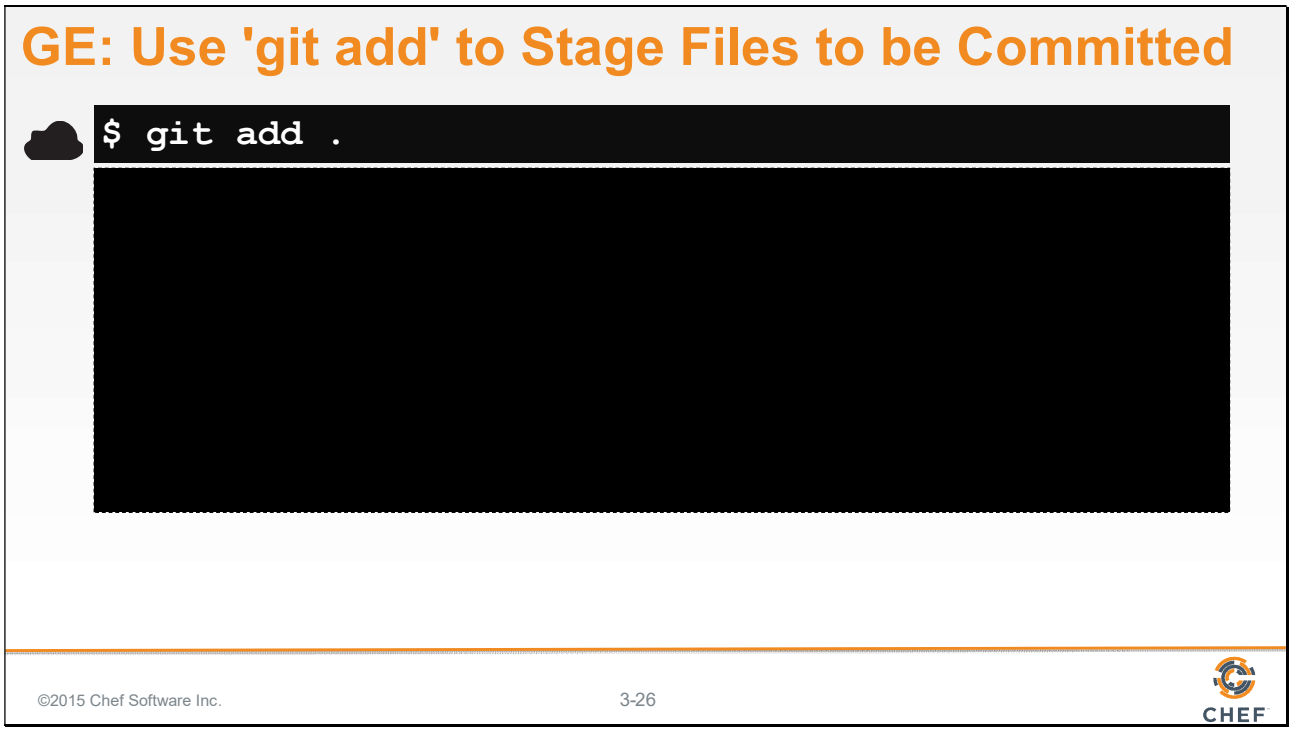
http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

CHEF

You can think of the staging area as a box in which to put a bunch of items -- like a care package you would send to someone.

Staging files means to put them in the box, but don't close it up because you may add a few things, and don't close it up because you may replace or remove a few things. But put the items in the box because eventually we are going to close that box when it is ready to send it off.

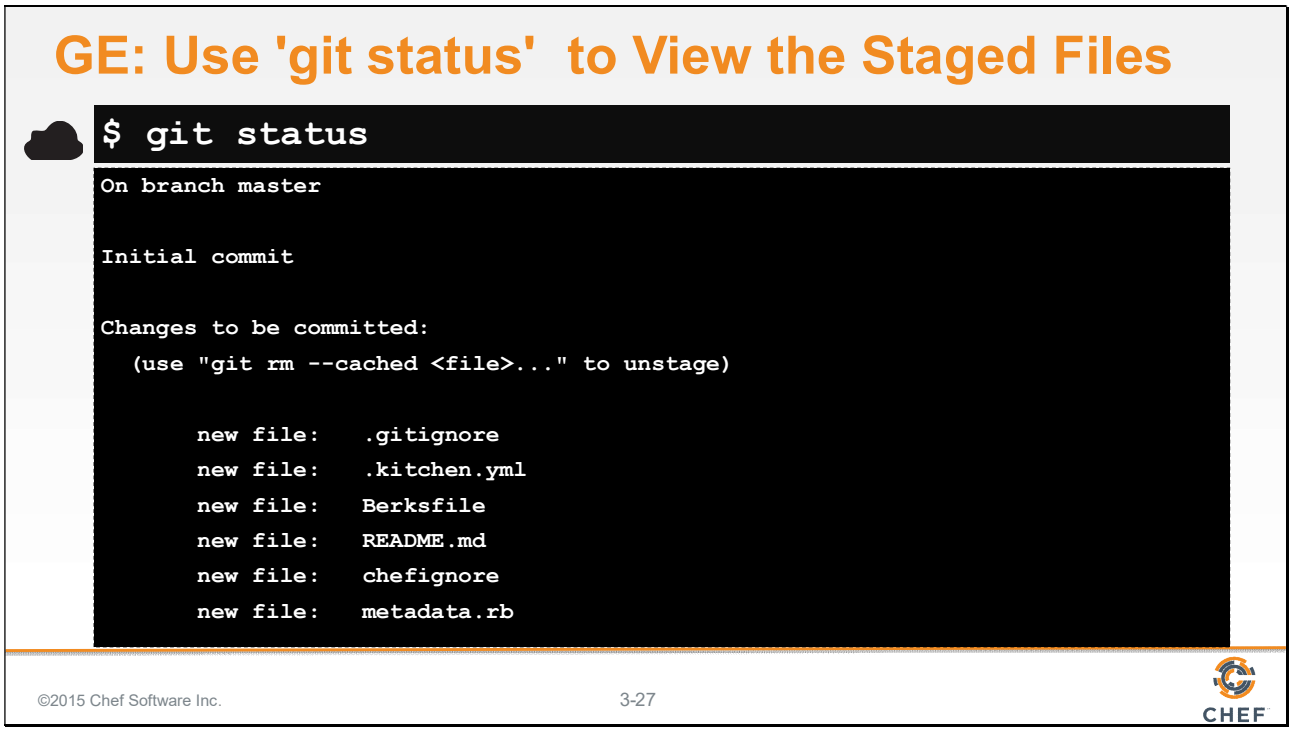
Slide 26



Now we need to tell git which files it should start tracking in source control. In our case, we want to add all the files to the repository and we can do that by executing '`git add .`' (dot).

This will place all the files into a staging area.

Slide 27



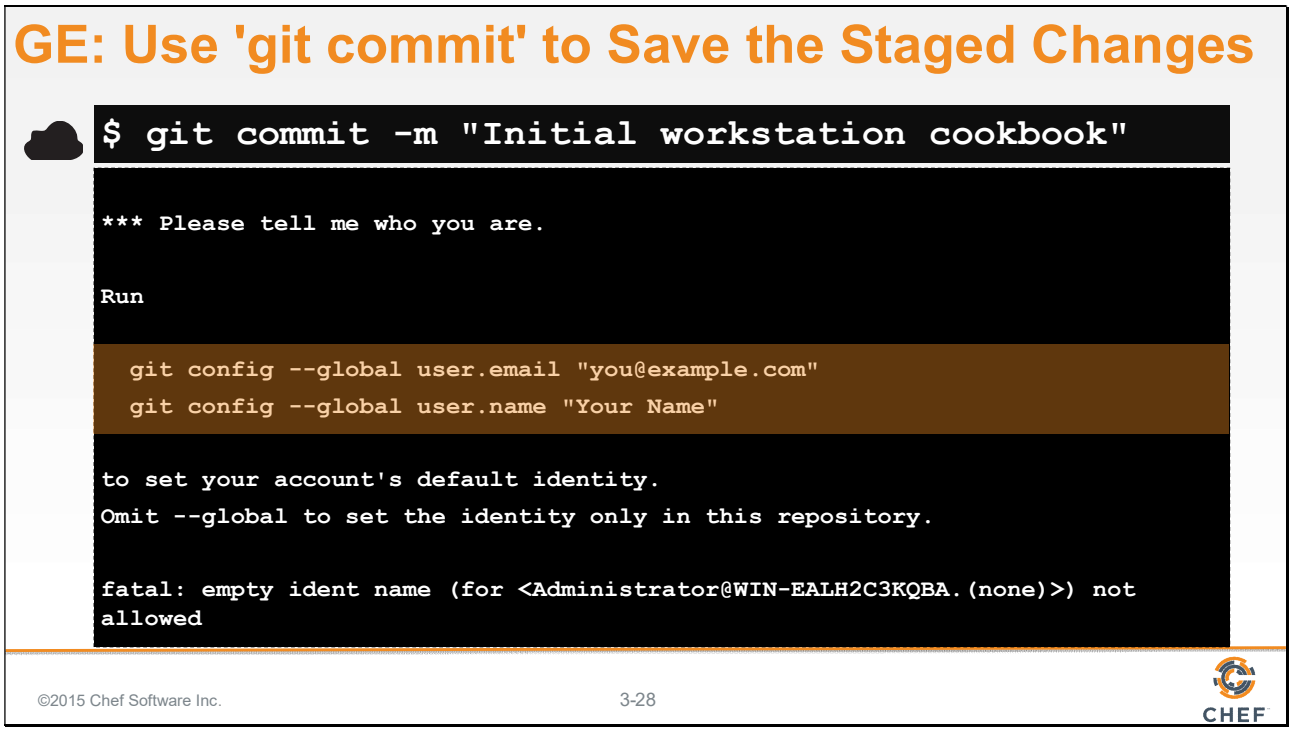
Let's see what changes we have placed in the staging area.

Thinking about our care package example, this is like looking inside the box and taking an inventory, allowing us to figure out if we need to move more things in or remove things we accidently threw in there.

Running `git status` allows us to see in the box. Git reports back to us the changes that will be committed.

Instructor Note: Git helpfully tries to show you the command you can use to remove an item from that box. This is useful if you want to include all items excepts for one or simply manage everything before you commit.

Slide 28



If everything that is staged looks correct, then we are ready to commit the changes.

This is like saying we're ready to close the box up.

This is done in git with **git commit**. We can optionally provide a message on the command-line and that is done with the **-m** flag and then a string of text that describes that change.

When we attempt to commit our changes git presents us with an error that states that you need to setup a user name and email address.

Slide 29

Set up your email you want to associate with your git commits.

Slide 30

Set up your user name you want to associate with your git commits.

If everything that is staged looks correct, then we are ready to commit the changes.

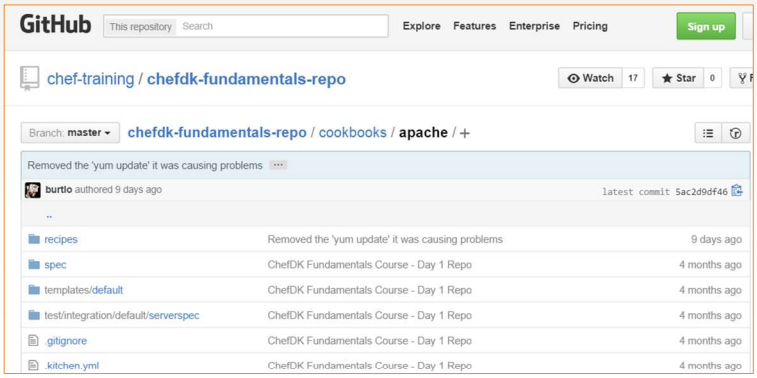
This is like saying we're ready to close the box up.

This is done in git with **git commit**. We can optionally provide a message on the command-line and that is done with the **-m** flag and then a string of text that describes that change.

Git Version Control

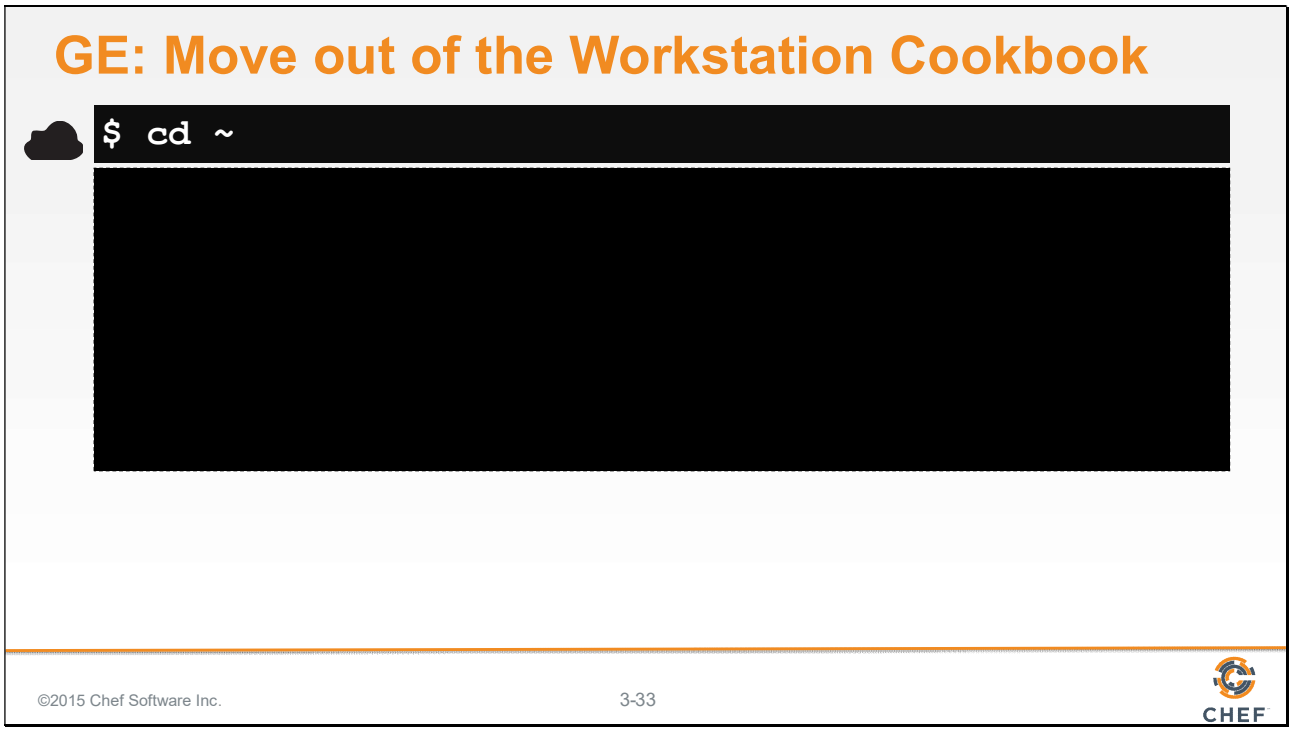
If you use git versioning you should ultimately push the local git repository to a shared remote git repository.

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



git tracks all our commits, all those closed up boxes, locally on the current system. If we wanted to share those commits with other individuals we would need to push those changes to a central repository where we could collaborate with other members of the team. GitHub is an example of a central git repository.

Slide 33



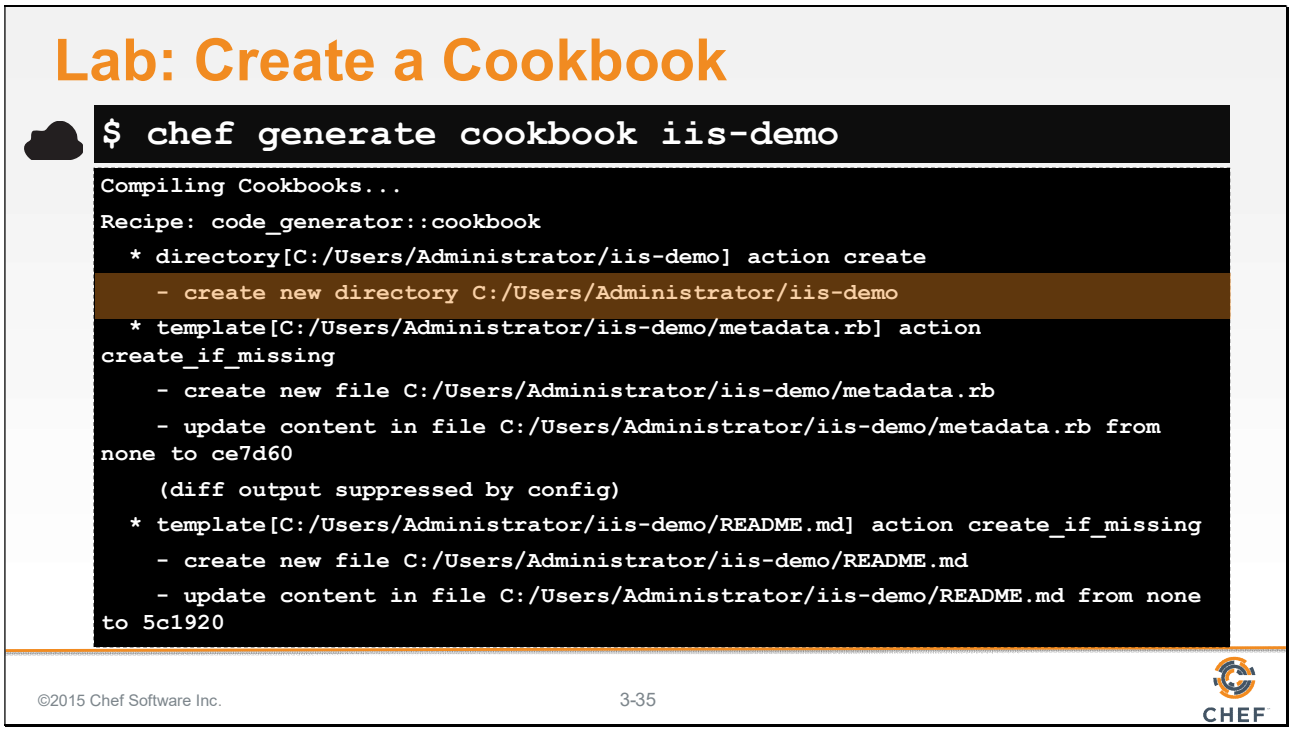
Now that we are done adding our workstation cookbook to version control lets return to our home directory.

Slide 34

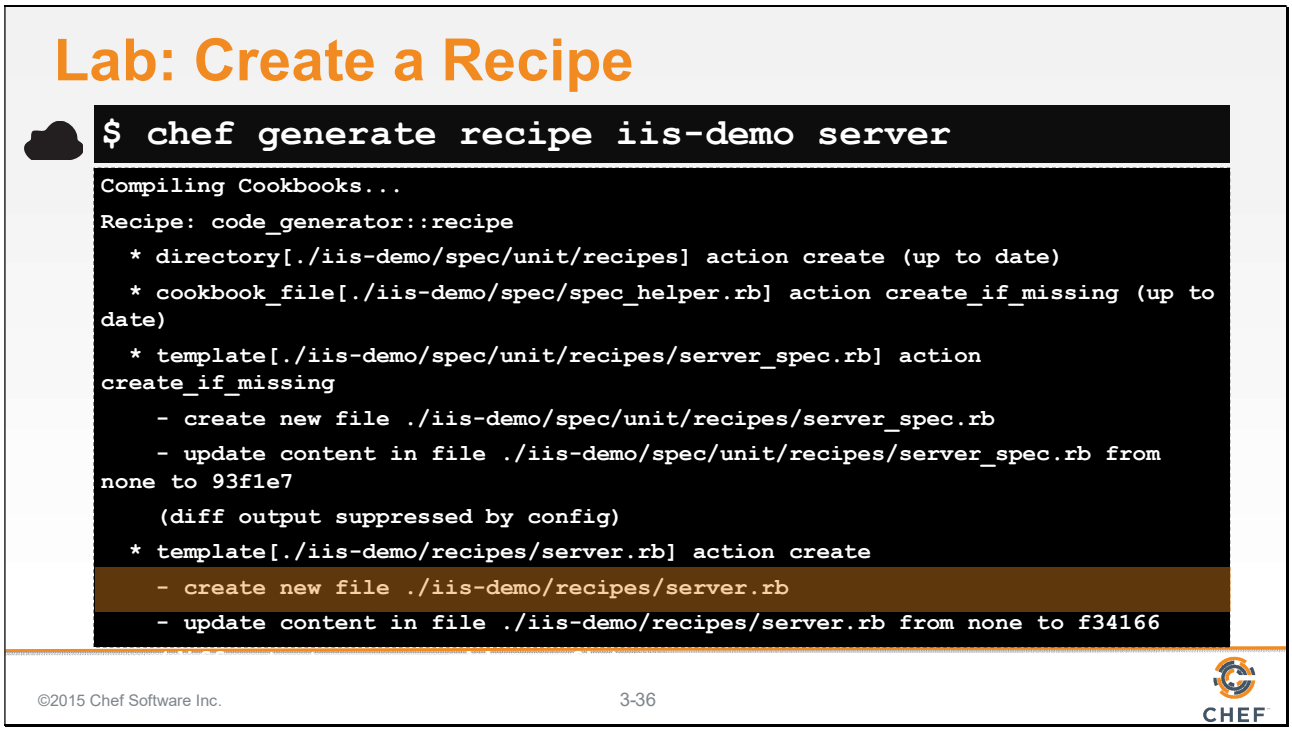
Now. Here is your last challenge: Deploying a Web Server with Chef.

We need a cookbook named iis-demo that has a server recipe. Within that server recipe we need to use a powershell_script to install the IIS windows feature, write out an example HTML file, and then start and enable the w3svc service.

Then we should apply that recipe and make sure the site is up and running by running a command to visit that site.



From the Chef home directory, run the command 'chef generate cookbook iis-demo'. This will place the iis-demo cookbook alongside the workstation cookbook.



From the Chef home directory, run the command 'chef generate cookbook iis-demo'. This will place the iis-demo cookbook alongside the workstation cookbook.

Slide 37

Lab: Create Server Recipe

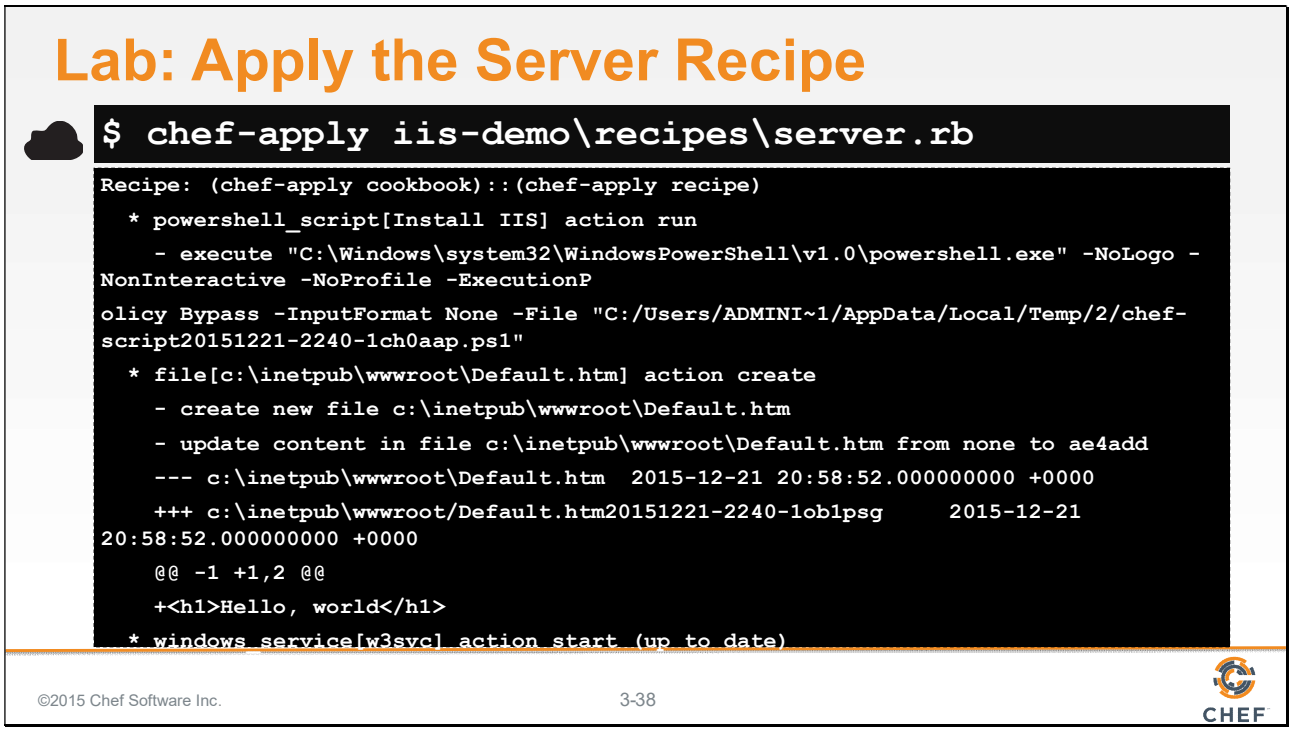
~\iis-demo\recipes\server.rb

```
powershell_script 'Install IIS' do
  code 'add-windowsfeature Web-Server'
end

file 'c:\inetpub\wwwroot\Default.htm' do
  content '<h1>Hello, world!</h1>'
end

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

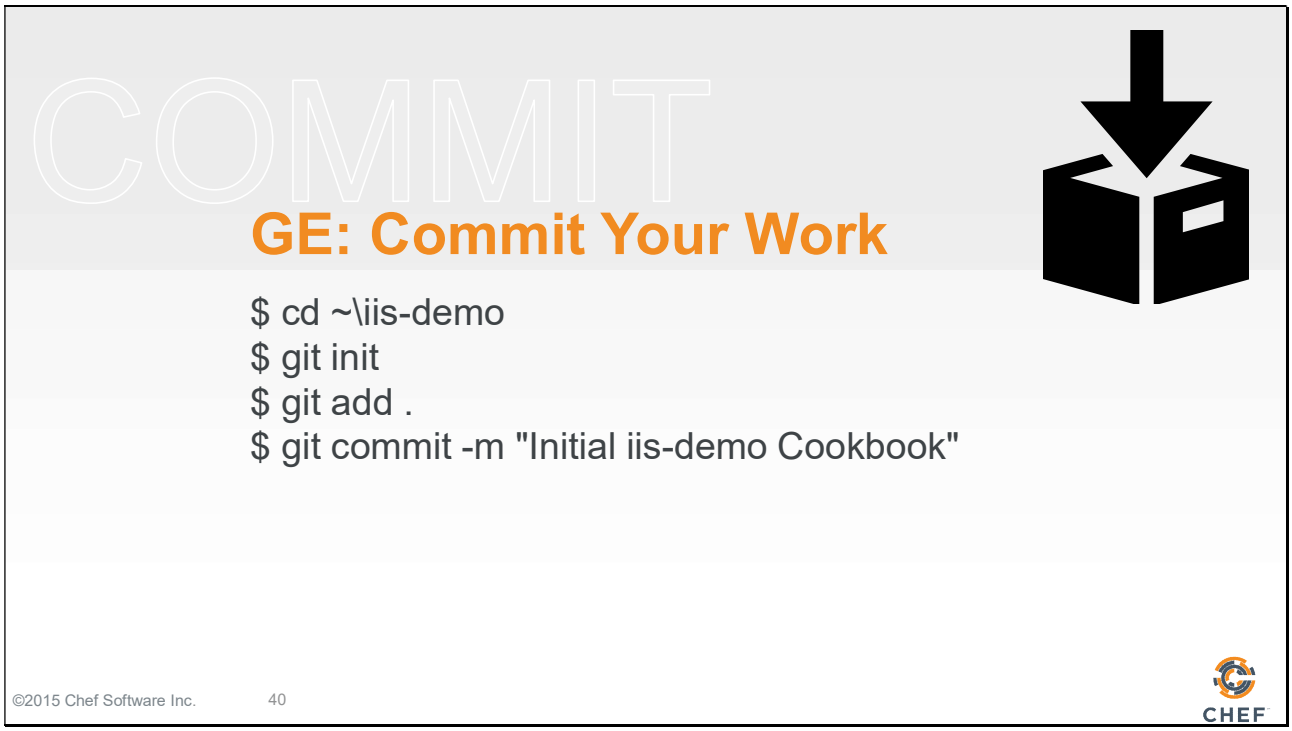

The server recipe, found at ~/iis-demo/recipes/server.rb, defines the policy:



When applying the recipe with '**chef-apply**', you need to specify the partial path to the recipe file within the iis-demo cookbook's recipe folder.

So verify that the website is available and returns the content we expect to see.

Slide 40



Now, with everything working it is time to add the iis-demo cookbook to version control.

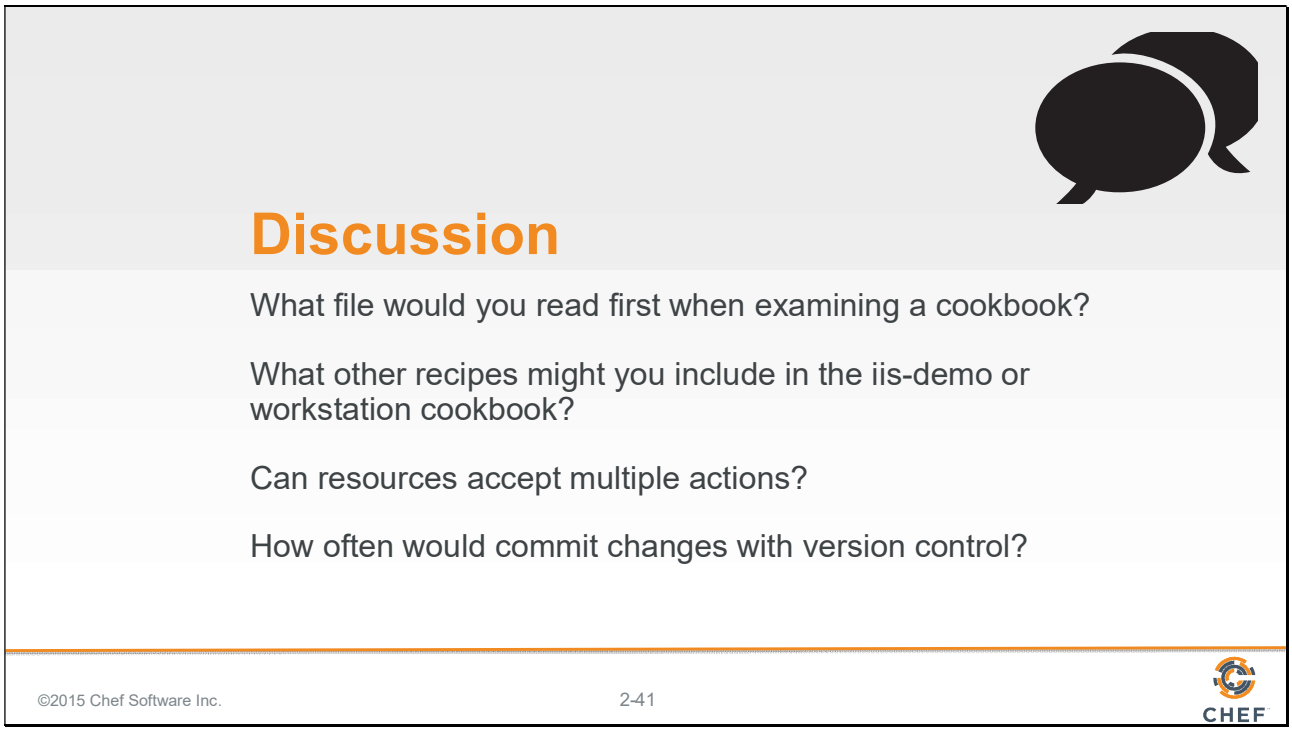
Move into the iis-demo directory.

Initialize the cookbook as a git repository.

Add all the files within the cookbook.

And commit all the files in the staging area.

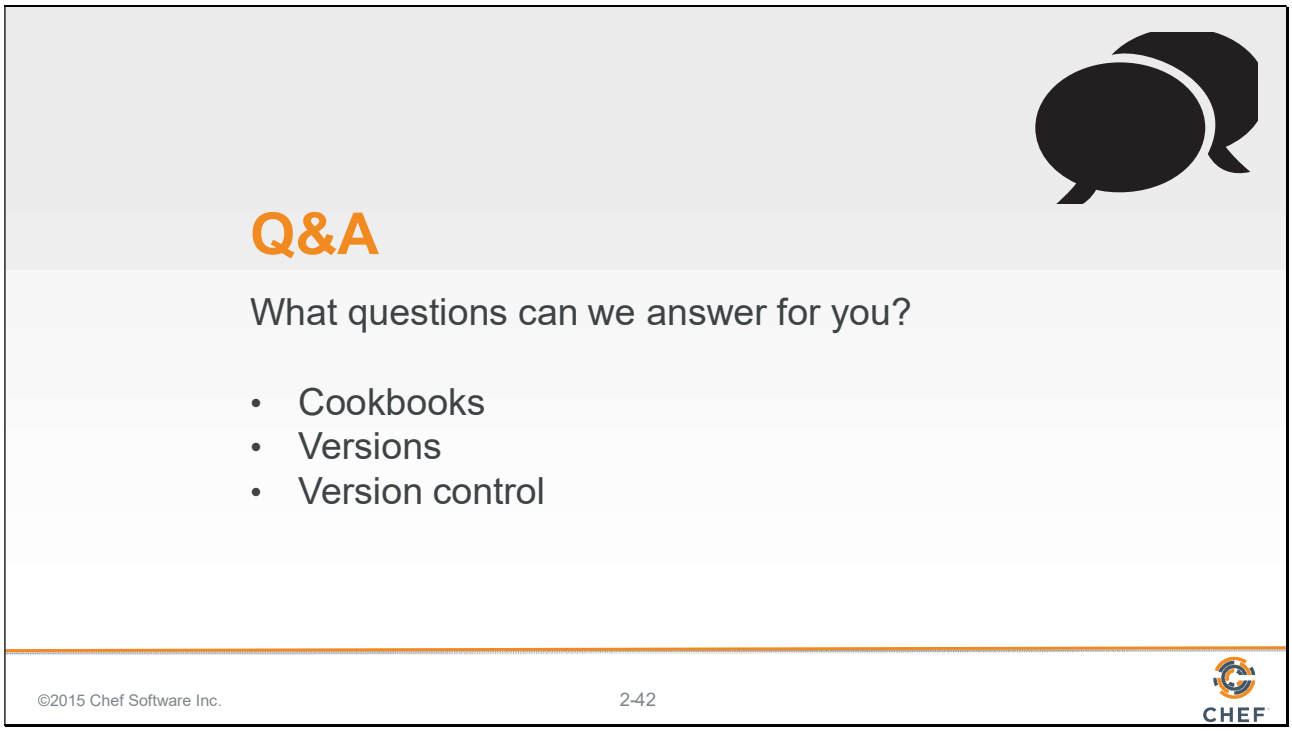
Slide 41



Answer these questions.

With your answers, turn to another person and alternate asking each other asking these questions and sharing your answers.

Slide 42



What questions can we help you answer?

General questions or more specifically about cookbooks, versioning and version control.

Slide 43