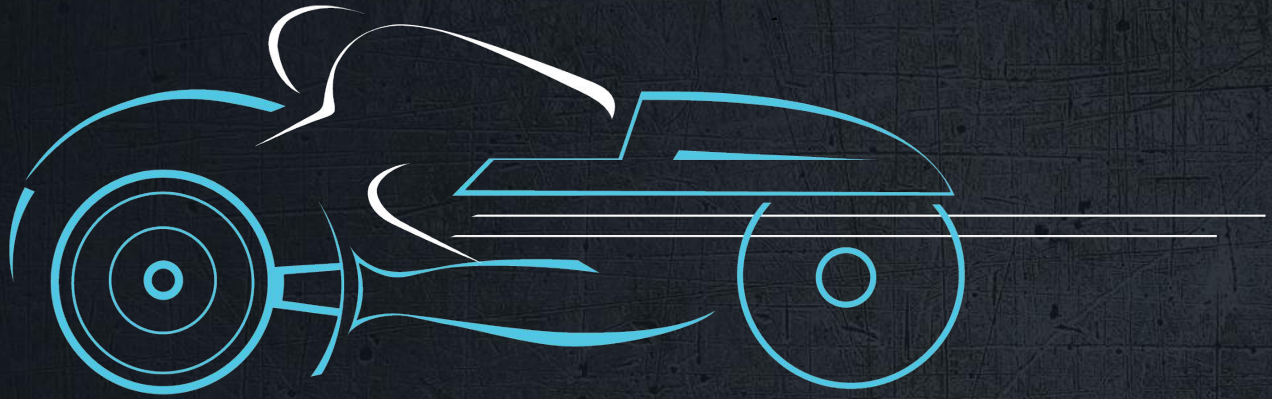

Ten Quick Tips to Improve Your Chef Workflow

Sean Carolan, May 3, 2015

CHEF

#1 Use the ChefDK

Contributor: Chef Software

The Chef Development Kit includes the knife command line tool, the new chef command, generators for cookbooks and other items, and testing tools like Rubocop, Foodcritic, Chefspec, and Test Kitchen.

#1 Use the ChefDK

Contributor: Chef Software

The Chef Development Kit includes the knife command line tool, the new chef command, generators for cookbooks and other items, and testing tools like Rubocop, Foodcritic, Chefspec, and Test Kitchen.

CHEF

#2 Windows Chef Workstation

Contributor: Adam Edwards

Install Chocolatey, ConEMU and PSReadLine and a good text editor.

Get a good text editor! Friends don't let friends use Notepad.

Install the ChefDK and always use Powershell, not CMD.exe

CHEF

#2 Windows Chef Workstation

Contributor: Adam Edwards

Install Chocolatey, ConEMU and PSReadLine and a good text editor.

Get a good text editor! Friends don't let friends use Notepad.

Install the ChefDK and always use Powershell, not CMD.exe

#3 ServerSpec Helper Files

Contributor: Joshua Timberman

When writing ServerSpec tests you can have a single shared config file for all your tests.

- Create a spec_helper.rb file to contain all your shared settings.
- Update the data_path setting in your .kitchen.yml file to read from the directory where it is stored.

#3 ServerSpec Helper Files

Contributor: Joshua Timberman

When writing ServerSpec tests you can have a single shared config file for all your tests.

- Create a spec_helper.rb file to contain all your shared settings.
- Update the data_path setting in your .kitchen.yml file to read from the directory where it is stored.

#4 Use MixLib::ShellOut

Contributor: Julian Dunn

- Now that shell_out and shell_out! are part of the recipe DSL in Chef 12, there's no excuse to abuse `backticks`.
- shell_out! will execute a command on the system and raise an error if the command fails.
- You can capture stdout and stderr with shell_out
- https://docs.chef.io/dsl_recipe.html#shell-out

#4 Use MixLib::ShellOut

Contributor: Julian Dunn

- Now that shell_out and shell_out! are part of the recipe DSL in Chef 12, there's no excuse to abuse `backticks`.
- shell_out! will execute a command on the system and raise an error if the command fails.
- You can capture stdout and stderr with shell_out
- https://docs.chef.io/dsl_recipe.html#shell-out

#5 Record While You Learn

Contributor: Nell Shamrell-Harrington

Looking over someone's shoulder while they teach you
their Chef workflow?

Ask them to capture their session using screen recording software. This way you can review everything they did later, command by command.

CHEF

#5 Record While You Learn

Contributor: Nell Shamrell-Harrington

Looking over someone's shoulder while they teach you
their Chef workflow?

Ask them to capture their session using screen recording software. This way you can review everything they did later, command by command.

CHEF

#6 Pre-baked VMs or Containers

Contributor: Ann Marie Fred

Spinning up a virtual machine or container and configuring it from scratch is expensive.

Pre-install large packages and files in your images so that your CI/CD pipeline can quickly spin up VMs for testing.

Leave smaller configuration details for Chef to take care of.

CHEF

#6 Pre-baked VMs or Containers

Contributor: Ann Marie Fred

Spinning up a virtual machine or container and configuring it from scratch is expensive.

Pre-install large packages and files in your images so that your CI/CD pipeline can quickly spin up VMs for testing.

Leave smaller configuration details for Chef to take care of.

#7 Troubleshoot with Pry

Contributor: Claire McQuin

Ever wish you could pause a Chef run and examine what it's doing more closely? The Ruby 'pry' gem allows you to do exactly this.

- Add this to the top of your recipe: require 'pry'
- Add this where you want your breakpoint: binding.pry
- Advance to the next resource with CTRL-D

CHEF

#7 Troubleshoot with Pry

Contributor: Claire McQuin

Ever wish you could pause a Chef run and examine what it's doing more closely? The Ruby 'pry' gem allows you to do exactly this.

- Add this to the top of your recipe: require 'pry'
- Add this where you want your breakpoint: binding.pry
- Advance to the next resource with CTRL-D

CHEF

#8 Use run_state to store data

Contributor: Michael Goetz

Sometimes we need to temporarily store data during a Chef run, but do not wish to store it in the node object.

The run_state is a perfect place to store temporary data and make it available to other resources in your Chef run.

https://docs.chef.io/recipes.html#node-run-state

CHEF

#8 Use run_state to store data

Contributor: Michael Goetz

Sometimes we need to temporarily store data during a Chef run, but do not wish to store it in the node object.

The run_state is a perfect place to store temporary data and make it available to other resources in your Chef run.

https://docs.chef.io/recipes.html#node-run-state

#9 Coerce everything instead of checking for nil

Contributor: Franklin Webber

Coerce everything instead of checking for nil

```
if param.nil?
  next
else
  ...
end
```

#10 Pimp My Text Editor

Contributor: Matt Stratton

When writing Chef code, use a good text editor with a project drawer or file browser.

Many text editors have plugins available for syntax highlighting, auto-completion, and improving your Ruby style.

You can even run lint and syntax checkers from within the editor.

CHEF™