CHEF®

Progress®

Chef Infra Client 16

Overview of new features and upgrade best practices.

10/2020

Executive Summary:

In addition to including a review of the top new features released in Chef Infra Client 16 and how to upgrade, this paper includes a history of key features released since Chef Infra 12 and other useful information for those planning to upgrade.

Table of Contents:

Overview of Key New Features

Chef Infra is a powerful automation platform that transforms infrastructure configuration into code. Whether you're operating in the cloud, on-premises, or in a hybrid environment, Chef Infra automates how infrastructure is configured, deployed, and managed across your network, no matter its size.

Chef Infra Client 16 is a smaller, faster version that is chock full of new resources, features, and supported platforms. What makes this set of releases particularly exciting is an unprecedented focus on streamlining the user experience for newcomers and seasoned veterans alike.

Infrastructure Automation Just Got Easier

With Chef Infra 16, your automation is only as complex as the problems you need to solve! In Chef Infra Client 16 we've put an emphasis on ensuring that our community and customers can create, customize, and update their Chef policy with as little friction as possible. Whatever your proficiency level or use case, we're dedicated to ensuring you have the tools you need to get up and running quickly.

[See what's new in Chef Infra 16.]

To that end, here is an overview of some of the features that make Chef Infra 16 the most robust release to date:

- **YAML Recipe Support**
- **Custom Resource Unified Mode**
- **Massive Cookstyle Improvements**
- **Additional Platform Support**
- **Significant Performance Enhancements**
- **New Resources and Resource Improvements**

YAML Recipe Support

Chef Infra features a robust library of declarative resources for configuring a wide variety of infrastructure components. Chef recipes can now be written in YAML, providing a simplified, low-code entry point to writing Chef Infra policy.

```yaml
---
resources:
  - type: "package"
    name: "httpd"
  - type: "template"
    name: "/var/www/html/index.html"
    source: "index.html.erb"
  - type: "service"
    name: "httpd"
    action:
      - enable
      - start
```

Custom Resource Unified Mode

One of Chef Infra's key strengths is its extensibility. In particular, custom resources allow you to define reusable logic for tasks unique to your organization or use case. Unified Mode allows resource authors to opt-in to a single phase of execution for Chef resources and Ruby code, streamlining the customization process for Ruby newbies and experienced pros alike.

Massive Cookstyle Improvements

Last fall we announced that Cookstyle had officially supplanted Foodcritic as the de facto linting tool for Chef Infra code. Since then we've kept up the momentum, adding nearly 200 new cops custom-tailored for Chef cookbooks. This includes automatic detection and correction of many common deprecations, and facilities for targeting specific client versions. The Chef Infra extension for Visual Studio Code includes built-in Cookstyle linting as well!

Additional Platform Support

We now support multiple Linux flavors on ARM (aarch64) including Ubuntu 20.04, Amazon Linux 2, RHEL 8, and SLES 15! Our full list of supported platforms is available here.

Significant Performance Enhancements

In Chef Infra 16 we launched a client that is up to 30% smaller on disk with performance optimizations to significantly speed up client runs on Microsoft Windows.

In Client 16.5 we continued to improve performance and greatly reduced the startup time of the chef-client process. Startup times on macOS, Linux, and Windows hosts are now approximately 2x faster than the 16.4 release.

New Resources and Resource Improvements

In addition to unified mode, functionality has been improved to reduce code duplication, control resource state, and define compile-time logic

OS-specific enhancements like plist, windows_security_policy, alternatives, and user_ulimit and improved Infra Client scheduling with chef_client_windows_task, chef_client_cron, and chef_client_systemd_timer

In Client 16.6 the chef_client_config resource was completed which was the final component needed to eliminate dependency on the chef-client cookbook.

Target Mode Improvements

In Client 16.6 we enabled the majority of Ohai plugins to execute remotely.
Now most Ohai data is available when using the target mode.

As with every major release, we've also added a ton of performance and functionality enhancements to Chef Infra 16.

For a full list of features & enhancement visit docs.chef.io.

To see a deeper dive of the new features launched in Chef 16 view the on-demand webinar

Support and Security Updates

The most immediate reason to keep clients updated is to maintain support and ensure the most up-to-date security patches. While new feature updates will be limited to the latest release, security patches and bug fixes will be provided for both during their support lifecycle.

Within a particular major release, Chef further recommends always running the latest version of that release. Updates published as minor or patch releases are as a rule intended to be non-breaking, backwards compatible, and most importantly, do not require updating associated cookbooks. These releases often feature performance improvements as underlying components are upgraded, additional platform support as new operating systems become available, and timely updates and patches in response to any nascent vulnerabilities or CVEs in any of Chef Infra's dependencies.

Chef provides support, bug fixes, and security patches for the latest two major releases of Chef Infra Client. That means that Chef Infra 15 will continue to be supported, but **Chef Infra 14 is officially End of Life** as of this release (April 28 2020).

Further information and a full list of software Chef supports can be found in our documentation.

Chef Infra Client Release History

Below is a summary of key features released from Chef Infra Client 12 to the current version Chef Infra 16. Note with the release of Chef Infra Client Chef 16, Chef Infra Client 12-14 are now end of live (EOL).

Chef Infra Client 12

Chef Infra Client 12 is unique in a number of ways. It was the final release before we formalized the yearly cadence of major releases, and was one of the longest running stable releases of Chef Infra. As such, a huge number of improvements were added during its lifecycle.

Release Highlights:

- **AIX Support Added**
- **Expanded Windows Support**
  - New Resources: windows_service, reboot, dsc_resource, chocolatey_package, cab_package, msu_package
  - 64-bit windows binaries
  - UNC path support in remote_file resource
- **Expanded macOS Support**
  - New Resources: homebrew_package, osx_profile
- **Other New Resources**
  - bff_package, openbsd_package, paludis_package, apt_update, launchd, yum_repository, ksh, systemd_unit
- **Removable Cookbook Dependencies**
  - Resources provided by the yum and systemd cookbooks are now natively implemented
- **Notification Timers**
  - Determine when a notifies or subscribes parameter is executed.
  - Supports :delayed (default), :before, :immediately
- **Security Updates**
  - Client/Server connections over HTTPS by default
  - FIPS Mode added
- Custom Resources **Introduced**
- Policyfiles **Introduced**
- **Chef Automate** data collection **Introduced**

Chef Infra Client 13

With Chef Infra Client 13, we established our current yearly major release cadence. Full details can be found in the Chef Infra Release and Support Schedule. As part of this change, any planned deprecations, syntax revisions, or other breaking changes must first be implemented as a non-breaking warning that indicates removal in the next major release. Similarly, while patches, bug fixes, and CVE remediations would continue to be implemented throughout each release, changes that might impact behavior or performance, like Ruby upgrades to the next minor release, would be scheduled for the next major release of Chef Infra Client.

Release Highlights:

- **New Resources**
  - apt_preference, windows_task, zypper_repository
- **Ohai Improvements**
  - Improved cloud support with expanded detection of EC2/Softlayer clouds and metadata gathering for Azure/Rackspace in Ohai
- **Removable Cookbook Dependencies**
  - apt
- **Encrypted Data Bags use more secure aes-256-gcm encryption method by default**
- **Chef InSpec and Chef Vault included by default**
- **Upgraded to Ruby 2.4**

Chef Infra Client 14

Chef Infra Client 14 saw a vast improvement in performance and reduction in install size. Additionally, we added a huge number of new resources that were previously provided by cookbooks on the Chef Supermarket. With these changes, Chef Infra practitioners not only saw the client itself become easier to manage, but could greatly reduce the number of cookbooks they needed to manage.

Release Highlights:

- **New Resources**
  - windows_workgroup, windows_shortcut, windows_printer_port, windows_printer, windows_font, windows_feature, windows_auto_run, windows_ad_join, sysctl, swap_file, sudo, rhsm_subscription, rhsm_repo, rhsm_register, rhsm_errata_level, rhsm_errata, openssl_rsa_public_key, openssl_rsa_private_key, openssl_dhparam, ohai_hint, macos_userdefaults, hostname, homebrew_tap, homebrew_cask, dmg_package, chef_handler, ssh_known_hosts_entry, kernel_module, powershell_package_source, chocolatey_source, chocolatey_config, openssl_ec_public_key, openssl_ec_private_key, openssl_x509_crl, openssl_x509_request, openssl_x509_certificate, cron_access, cron_d, windows_workgroup, locale, timezone, windows_firewall_rule, windows_share, windows_certificate, and build_essential

- **Improved Resources**
  - Windows_service can now create Windows services
  - Large improvements to yum package installation
- **Removable Cookbook Dependencies**
  - windows, build_essential, mac_os_x, openssl, sudo, sysctl, rhsm, homebrew, windows_firewall, swap, hostname-chef, locale, timezone_iii
- **Expanded Platform Support**
  - MacOS 10.14 (Mojave), SLES 15, Windows 2019, Windows 10, FreeBSD 12, AIX 7.2, and RHEL 8
- **Improved FIPS detection**
- **Install size reduced by 50% on Linux/macOS, 12% on Windows**
- **Upgraded to Ruby 2.5**

Chef Infra Client 15

Chef Infra Client 15 is currently supported, and will remain so through April 2021. It also coincided with an update to our licensing policies, in which we made all of Chef's software open source under an Apache2 license, and their supported distributions (binaries) subject to an enterprise license for commercial use. More detail can be found in this blog post from February. Additionally, this release featured a significant number of new helper functions to help with cookbook creation and the first phase of expanded ARM support that continued in Chef Infra 16.

Release Highlights:

- **New Resources**
  - snap_package, archive_file, windows_uac, windows_dfs_folder, windows_dfs_server, windows_dns_record, windows_dns_zone, chocolatey_feature, chef_sleep, notify_group
- **New Helpers to simplify writing cookbooks and resources**
  - Multiple platform detection helpers for cloud, virtualization, and OS version
  - include_recipe? enables conditional execution based on other recipes in use
- **Removable Cookbook Dependencies**
  - windows_dfs, windows_dns, libarchive
- **Expanded Platform Support**
  - x86_64: Ubuntu 20.04, Debian 10, macOS 10.15 (Catalina), Amazon Linux 2
  - aarch64: Ubuntu 18.04, RHEL 7, Amazon Linux 2, SLES 15
- **Support for Ed25519 SSH keys**
- **Unified Bootstrapping of *nix/Windows**

- **Target Mode introduced**
  - provides platform-agnostic configuration over SSH
- **Upgraded to Ruby 2.6**

Chef Infra Client 16

Chef Infra Client 15 is currently supported, and will remain so through April 2021. It also coincided with an update to our In addition to to the key capabilities discussed earlier like YAML Recipes and a unified execution mode for custom resources, Chef Infra 16 has a ton of additional features, including:

- **Added YAML Recipe Support**
- **Expanded ARM support**
  - aarch64 builds for RHEL8, Ubuntu 20.04, and SLES16
- **Reduced disk usage by up to 30% with drastically improved performance on Windows systems**
- **Windows improvements**
  - PowerShell Core Support
  - Enhanced 32-bit Windows Support
- **8 new resources**
  - alternatives, plist, user_ulimit, windows_security_policy, windows_user_privilege, chef_client_cron, chef_client_systemd_timer, chef_client_scheduled_task
- **Improvements to 10 existing resources**
  - build_essential, cron, dnf_package, git, locale, msu_package, package, service, windows_firewall_rule, windows_package
- **2 new helper functions can be used in any resource or recipe**
  - sanitized_path, which
- **Ohai Plugin Improvements**
  - Improved Azure & Linux Network data
  - New plugins for IPC and Interrupts
  - DMI plugin support for Windows
- **Custom Resource Improvements**
  - Unified Mode (single phase execution)
  - Improved property require behavior
  - Resource partials for code reuse between resources
  - New after_resource state

    - Improvements and default behavior for identity and desired_state properties
- **The compile_time property is now available for all resources, including custom resources**
- **Upgraded to Ruby 2.7**

Upgrade Preparation Useful Info

There's a lot of ground to cover when we talk about upgrades, and there are a lot of important concepts that don't necessarily fit into some of the more specific topics we have in store. With that in mind, we'll start with an overview of some key concepts that should prove invaluable going forward.

Chef Infra Server Versions

An important thing to note up front is that the Chef Infra Client follows a yearly major release schedule, whereas the Chef Infra Server does not. This structure allows us to continuously improve Chef Infra while providing a predictable timeline for any planned feature updates or breaking changes, but does mean that there is not parity between client and server version numbers.

As of this writing, the latest stable Chef Infra Server release is 13.2.0, and Chef provides support for 12.x and 13.x releases. Either version can be used regardless of the Chef Infra Client version you have in place.

Chef Workstation & ChefDK

Chef Workstation was introduced two years ago, and is intended to act as a drop-in replacement for the ChefDK. While Chef still supports and maintains ChefDK, Workstation contains the same tools, alongside additional enhancements like ad-hoc task support with chef-run and a desktop application with auto-update facilities.

Also of note, regardless of what version of Chef Infra you're using, Chef recommends running the latest version of Chef Workstation. Tools like Cookstyle and Test Kitchen provide built-in facilities for targeting specific Chef Infra versions, so upgrading workstation will not require you upgrade Infra Client as well.

Upgrading Cookbooks & Clients

Upgrading Chef Infra Client is really a twofold process:

- **Updating Chef cookbooks to ensure compatibility**
- **Upgrading Chef Infra Client itself on managed systems**

How much planning and work is required for each task will vary from organization to organization. Some may find that their cookbooks need very little work, and can focus their efforts on orchestrating client upgrades. Some will have large numbers and combinations of cookbooks in place across environments and platforms, and look to subdivide the upgrade process to keep the process manageable. In this series, we'll address these tasks individually, and provide tips & tricks for each.

Upgrading to Chef Infra 16

The goal of this section is to provide practical advice on how to go about upgrading a Chef Infra Client. Specifically, we'll be covering how to evaluate whether your cookbooks will be compatible with the latest versions, and how to use the tools provided in Chef Workstation to quickly iterate on and test those cookbooks against a target release.

Upgrading Cookbook Guidance and FAQ

This guidance is specific for upgrading your cookbooks which is often done in conjunction with a major client upgrade.

Chef Upgrade Lab

Chef has taken its learnings from helping clients upgrade our cookbooks and clients at scale, with the new Upgrade Lab. Upgrade Lab provides a way to evaluate every cookbook and node on your Chef Infra Server to ensure a straightforward path to migrating everything to the latest and greatest kit.

How do I know which issues I'm affected by?

Great question! In the past you needed to download multiple versions of ChefDK that mapped to the Chef Infra Client version you were going from, to the version you were going to. Then you'd run foodcritic against each cookbook and start traipsing through the output.

With the introduction of Chef Cookstyle, that's not a thing anymore! We can simply set a target version in our cookbook's .rubocop.yml (in this example we'll be going from 12 to 13) and get rolling! It's important to note when you specify a version constraint like we've done, that only the warnings specific to this version will display. When you change the target version or release the specification, you'll have more to fix. Please check out the Chef Cookstyle documentation for a complete review!

If you'd like to try an upgrade on a test repository to see some errors, first clone this repo (github.com/ChefRycar/bad_dates) and make sure you have the latest version of Chef Workstation installed (downloads.chef.io/chef-workstation/).

Which offenses do I fix first?

The ones that Cookstyle can autocorrect of course! What's that? Cookstyle has an autocorrect feature in it? Yes it does. We'll get to that shortly, but first we want to see which deprecations exist in our cookbook and then we'll perform a frequency analysis.

To check offenses that exist when upgrading from Chef Infra Client 12 to 13.latest:

```
cd ~/path/to/bad_dates
```

```
> cookstyle .
```

You'll now see output that looks like this:

```
 /tmp/bad_dates# cookstyle .
Inspecting 9 files

.WWWWW...
```

Offenses:

```
metadata.rb:8:1: R: ChefSharing/InsecureCookbookURL: Insecure http Github or Gitlab URLsfor
metadata source_url/issues_url fields

issues_url 'http://github.com/ChefRycar/bad_dates/issues'

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

*snip*

recipes/windows.rb:13:10: W: ChefDeprecations/ChocolateyPackageUninstallAction: Use
the:remove action in the chocolatey_package resource instead of :uninstall which was
removedin Chef Infra Client 14+

action :uninstall

^^^^^^^^^^
```

9 files inspected, 18 offenses detected

Now let's do a quick frequency analysis:

```
/tmp/bad_dates# cookstyle . --format offenses

9/9 files |============================================== 100==============================
===============>| Time: 00:00:00

3   ChefDeprecations/ChefRewind

3   ChefDeprecations/DeprecatedYumRepositoryProperties

2   ChefSharing/InsecureCookbookURL

1   ChefCorrectness/NodeNormal

1   ChefDeprecations/ChefWindowsPlatformHelper

1   ChefDeprecations/ChocolateyPackageUninstallAction

1   ChefDeprecations/CookbookDependsOnCompatResource

1   ChefDeprecations/CookbookDependsOnPartialSearch

1   ChefDeprecations/CookbookDependsOnPoise

1   ChefDeprecations/EpicFail

1   ChefDeprecations/NodeSet
```

```
1   ChefDeprecations/WindowsTaskChangeAction
1   Style/StringLiterals
--

18 Total
```

Now that's useful information! At this point if we really want to dive in, we can compare the output from the analysis with all of the rules that currently exist in Cookstyle (github.com/chef/cookstyle/blob/master/docs/cops.md).

For the sake of brevity, we won't do that here. Instead we'll look at the autocorrect logic.

To use Cookstyle to autocorrect your cookbook:

```
/tmp/bad_dates# cookstyle -a .
Inspecting 9 files

.WWWWW...
```

Offenses:

```
metadata.rb:8:1: R: [Corrected] ChefSharing/InsecureCookbookURL: Insecure http Github
orGitlab URLs for metadata source_url/issues_url fields

issues_url 'http://github.com/ChefRycar/bad_dates/issues'

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

metadata.rb:9:1: R: [Corrected] ChefSharing/InsecureCookbookURL: Insecure http Github
orGitlab URLs for metadata source_url/issues_url fields

source_url 'http://github.com/ChefRycar/bad_dates'

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

*snip*

recipes/windows.rb:9:11: W: [Corrected] ChefDeprecations/WindowsTaskChangeAction: The:change
action in the windows_task resource was removed when windows_task was added toChef Infra
Client 13+. The default action of :create should can now be used to create an update tasks.

 action [:change, :create]

       ^^^^^^^

recipes/windows.rb:13:10: W: [Corrected]ChefDeprecations/ChocolateyPackageUninstallAction:
Use the :remove action in the chocolatey_package resource instead of :uninstall which was
removed in Chef Infra Client14+

 action :uninstall

      ^^^^^^^^^^
```

9 files inspected, 18 offenses detected, 15 offenses corrected

From 18 offenses down to 3 in only a few seconds! This will clearly scale.

Incredibly important note here:

Once you've used Cookstyle's autocorrect feature, you should absolutely, without exception, run your cookbook through your full suite of local and integration testing using tools like Test Kitchen and its integration in your CI/CD pipeline.

Will correcting these offenses break my current chef-client runs and/or pipeline build runners?

This is by far the most common and important question our customers ask us with respect to this process. It is Chef's opinion that when working through the cookbook version and client upgrades, you perform them one major version at a time. In this example, we're moving from 12.x compatibility to 13.latest compatibility. By doing this you're introducing code changes required in 13 that are supported in 12, thus giving you the confidence and control to upgrade cookbook by cookbook instead of all at once. Once your codebase has full compatibility with version 13, you can seamlessly upgrade the Chef Infra Client version running in your estate.

Once you're completely stable on the version of Infra Client that you just moved to, you repeat the process from 13 => 14, 14 => 15, and finally 15 => 16. A quick and easy way to do this is by modifying the TargetChefVersionattribute in the .rubocop.yml in your cookbook to the version of the client you're intending to upgrade to.

Can I upgrade from Chef Infra Client 12 to 16?

Yes, you can. But because you'll no longer have backward compatibility with your codebase, you'll need to introduce the new code and new client version in an extremely controlled fashion. This can be done, but isn't advised unless thoughtfully planned out and vetted by one of Chef's Customer Architects.

How should I release these changes in a controlled fashion?

Let's continue with the sample repo provided. By now you should still have 2 offenses present in the bad_dates example cookbook.

To take a look at how to correct what's left, run the following command without the offenses formatter and make the necessary changes:

```
/tmp/bad_dates# cookstyle .
Inspecting 9 files

.WR.....
```

Offenses:

```
metadata.rb:12:1: W: ChefDeprecations/CookbookDependsOnPartialSearch: Don't depend on the deprecated partial_search cookbook made obsolete by Chef 13

depends 'partial_search'

^^^^^^^^^^^^^^^^^^^^^^^^

recipes/default.rb:9:1: R: ChefCorrectness/NodeNormal: Do not use node.normal. Replacewith default/override/force_default/force_override attribute levels.
```

```
node.normal['antipattern'] = true
^^^^^^^^^^^
```

9 files inspected, 2 offenses detected

You'll now see we have a couple of offenses that weren't autocorrected by Cookstyle. However, you'll always be given a path forward or an explanation for what's presented. In this case, we see partial_search is remediated by moving to Chef Infra Client 13, which is exactly what we're doing! We also see a warning against using node.normal and should defer to our attribute precedence documentation to choose a way forward.

Once the offenses are corrected and after a series of successful integration tests, you have a cookbook that is completely compatible with Chef Infra Client 13. Above all, it's backwards compatible with your existing Chef Infra Client 12 nodes. You can now begin releasing this cookbook into your pre-production environments with the goal of promotion to your production environment. After your cookbook code is sorted, you may now upgrade the Chef Infra Client itself to the target version. Rinse and repeat until you're at Chef's latest kit!

How can I handle this in the future?

This sounds like an insurmountable challenge. Like all things in life, you get better with practice. The goal should be to never let yourself get more than 1 major version behind what is currently supported by Chef. This starts with shoring up how you perform local development.

Once you're on Chef Client 15/16 and the latest Chef Workstation, you can use Test Kitchen attributes (https://docs.chef.io/workstation/config_yml_kitchen/) to be declarative about the versions you want to write code for. Adding the following to your kitchen.yml will ensure that the cookbook is compiled on the target version, as well as any deprecations will present as errors, forcing the developer to keep their code compatible.

In your kitchen.yml:

Set target version (latest by default)
https://docs.chef.io/workstation/config_yml_kitchen/#new-provisioner-settingsproduct_version: latest

Set deprecations as errors (https://docs.chef.io/workstation/config_yml_kitchen/#provisioner-settings)

```
provisioner:
  deprecations_as_errors: true
```

You can also add Cookstyle autocorrection to run automatically in local development, or add it into your CI/CD pipeline's linting/correction phase.

Using these simple methods will enable you to correct deprecations and offenses as part of the natural lifecycle of your codebase, rather than allowing it to become a monumental effort.

Upgrading Chef Infra Client Guidance

Once you've updated your cookbooks, and ensured compatibility with Test Kitchen, it's time to start upgrading clients in your environments. Before starting any upgrade process, be sure to check out the upgrade considerations in our documentation.

After the chef-infra-client upgrade, we will see the following benefits:

- **Vulnerability and Security patches (years of patched vulnerabilities and exposures)**
- **Utilization of a supported release**
- **Access new features and resources: YAML, additional resources, blah, etc.**
- **Bug fixes and general speed improvements**

The Manual Path

Upgrading something that potentially manages your entire infrastructure can appear burdensome. While there are more automated ways of dealing with this task, it's important to understand how we would approach it manually. The manual way of upgrading the chef-infra-client involves running a CLI command to upgrade on a node-by-node basis.

The following line would do the trick on *nix & macOS:

```
curl -L https://chef.io/chef/install.sh | sudo bash
```

And here's the Windows one-liner:

```
. { iwr -useb https://omnitruck.chef.io/install.ps1 } | iex; install
```

While having to SSH or WinRM into machines and upgrading via command-line is not extremely challenging, we can run into several problems:

- **Specifying different Chef Infra Client versions**
- **Dealing with different native OS processes**
- **Time costs of upgrading at scale**

For these very reasons, we've created the chef_client_updater cookbook, which does most of the heavy lifting for you.

The Accelerator chef_client_updater

The chef_client_updater allows you to upgrade your Chef Infra Client through a cookbook.
This recommended approach leads to an easy to digest process which gives you the following benefits:

- **Upgrading to any desired version**
- **In general, binaries expect to be upgraded one minor release at a time. This is generally done so functionality remains throughout the upgrade process.**
- **We've already taken care of functionality through the Cookstyle/refactoring exercise, which means we can jump to whatever major version we desire.**

- **Through the chef_client_updater we can simply pin the version of Chef Infra Client we want to migrate to and press play.**
- **Upgrading at Scale**

Your current Chef Infra consumption may translate to the challenge of having to upgrade hundreds or thousands of nodes. Thankfully Chef lives for Automation!

If you are already utilizing Chef Infra at that level of scale, then your nodes are probably already communicating with some version of the Chef Infra Server. By adding the chef_client_updater cookbook to your run-list and pushing it to the Chef Infra Server, your nodes will auto-upgrade the chef-client the next time they self-check for desired states.

This upgrade will only happen if the Chef Infra Client installed version doesn't match your desired version.
This means that the setup process only needs to happen once, and Chef Infra handles the rest.

The entire process is fairly fast and after you execute it once, you can apply it to whatever scale you want. Simply add the chef_client_updater as one of your dependencies and use the resource like so:

```
chef_client_updater 'Install latest Chef Infra Client 16.x' do
version '16'

end
```

Here's a video of performing a quick upgrade on a Windows machine:

The Modern Path: Effortless Configuration Pattern

Throughout the years Chef and the Chef Community have released several patterns of implementing Chef Infra. These patterns have tackled problems ranging from dependency handling to desired-state simplification. From the Berkshelf way to environment cookbooks and Policyfiles.

The effortless configuration pattern utilizes Policyfiles and Chef Habitat as the vehicle in which we deliver infrastructure. Why Chef Habitat? Habitat is a framework which allows one to "define, package, deliver" any project along with all of its dependencies, both immediate and transitive, in a consistent manner.

Organizations are complex and different teams might rely on different versions of chef-client. Some teams might be okay with their current setup for the time being, and might not be ready for any modifications just yet. Chef Habitat bundles the Chef Infra Client with configuration policy as a single artifact, where they can be promoted through release channels providing fine-grained control of where each version of client and policy should be in use.

With this approach, Policyfiles will still define your desired-state in an easy to read single-file definition and Habitat will be responsible for packaging the Policyfiles along with all the Chef Infra requirements (e.g. chef-infra-client, chef-scaffolding,etc.).

Why is this being brought up?

Effortless Config will create an immutable artifact out of your desired-state with a pinned chef-infra-client version.

Team X and Y can work with artifact A which uses an older version of Chef Infra, while Team Z works with the latest and greatest artifact B. Both teams could utilize the same pipeline without obstructing each other. This pattern erases any agent management problem that you might have previously encountered. The effortless config pattern also results in faster deployments/remediation at scale due to Habitat's lightweight ability to continuously communicate with the artifact repository.

If you're interested in learning more about the effortless configuration pattern, be sure to check out our documentation for details, caveats, and usage examples. We've also released a newly-redesigned Chef Infra course at Learn Chef that provides a detailed introduction to the pattern.

Additional Resources

- **Chef Infra Client Upgrade Documentation**
- **The chef_client_updater Cookbook**
- **An overview of the Effortless Config pattern**
- **The newly-relaunched Learn Chef experience**

As ever, if you'd like to upgrade but could use assistance, be sure to contact us to see how we can help jumpstart your upgrade process!

ABOUT CHEF

Chef is the global leader in DevSecOps and the developer of Chef Enterprise Automation Stack™ automating infrastructure, compliance and application delivery for more than half of the Fortune 500. For more than 10 years, Chef has led the industry in DevOps innovation, uniting teams at organizations of all sizes and optimizing processes and outcomes to accelerate its customers' business growth. Chef Software is developed as 100 percent open source under the Apache 2.0 license.

For more visit http://www.chef.io