



Filter by product and version

Chef: current



Overview

Chef

Getting Started

Tutorials

Concepts

Setup

Cookbook Reference

About Cookbooks

Attributes

Definitions

Files

Libraries

Recipes

Recipe DSL

Resources

About Resources

Common Functionality

Custom Resources

All Resources (Single Page)

apt_package

apt_repository

apt_update

bash

batch

bash ¶

[\[edit on GitHub\]](#)

Use the **bash** resource to execute scripts using the Bash interpreter. This resource may also use any of the actions and properties that are available to the **execute** resource. Commands that are executed with this resource are (by their nature) not idempotent, as they are typically unique to the environment in which they are run. Use `not_if` and `only_if` to guard this resource for idempotence.

Note

The **bash** script resource (which is based on the **script** resource) is different from the **ruby_block** resource because Ruby code that is run with this resource is created as a temporary file and executed like other script resources, rather than run inline.

Syntax ¶

A **bash** resource block executes scripts using Bash:

```
bash 'extract_module' do
  cwd ::File.dirname(src_filepath)
  code <<-EOH
    mkdir -p #{extract_path}
    tar xzf #{src_filename} -C #{extract_path}
    mv #{extract_path}/*/* #{extract_path}/
  EOH
  not_if { ::File.exist?(extract_path) }
end
```

where

- `cwd` specifies the directory from which the command is run
- `code` specifies the command to run

The full syntax for all of the properties that are available to the **bash** resource is:

```
bash 'name' do
  code           String
  creates        String
  cwd            String
  environment    Hash
  flags          String
  group          String, Integer
  notifies       # see description
  path           Array
```

bff_package

breakpoint

cab_package

chef_gem

chef_acl

chef_client

chef_container

chef_data_bag_item

chef_data_bag

chef_environment

chef_group

chef_handler

chef_mirror

chef_node

chef_organization

chef_role

chef_user

chocolatey_package

cookbook_file

cron

csh

deploy

directory

dpkg_package

dsc_resource

dsc_script

easy_install_package

env

```

provider
returns
subscribes
timeout
user
umask
action
end

```

```

Chef::Provider::Script:
Integer, Array
# see description
Integer, Float
String, Integer
String, Integer
Symbol # defaults to :r

```

where

- bash is the resource
- name is the name of the resource block
- cwd is the location from which the command is run
- :action identifies the steps the chef-client will take to bring the node into the desired state
- code, creates, cwd, environment, flags, group, path, provider, returns, timeout, user, and umask are properties of this resource, with the Ruby type shown. See “Properties” section below for more information about all of the properties that may be used with this resource.

Actions ¶

This resource has the following actions:

:nothing

Prevent a command from running. This action is used to specify that a command is run only when another resource notifies it.

:run

Default. Run a script.

Properties ¶

This resource has the following properties:

code

Ruby Type: String

A quoted (” ”) string of code to be executed.

creates

Ruby Type: String

Prevent a command from creating a file when that file already exists.

cwd

Ruby Type: String

<code>erl_call</code>	The current working directory.
<code>execute</code>	<u>environment</u> Ruby Type: Hash
<code>file</code>	A Hash of environment variables in the form of <code>({"ENV_VARIABLE" => "VALUE"})</code> . (These variables must exist for a command to be run successfully.)
<code>freebsd_package</code>	
<code>gem_package</code>	<u>flags</u> Ruby Type: String
<code>git</code>	One or more command line flags that are passed to the interpreter when a command is invoked.
<code>group</code>	
<code>homebrew_package</code>	<u>group</u> Ruby Types: String, Integer
<code>http_request</code>	The group name or group ID that must be changed before running a command.
<code>ifconfig</code>	
<code>ips_package</code>	<u>ignore_failure</u> Ruby Types: TrueClass, FalseClass
<code>ksh</code>	Continue running a recipe if a resource fails for any reason. Default value: <u>false</u> .
<code>launchd</code>	
<code>link</code>	<u>notifies</u> Ruby Type: Symbol, 'Chef::Resource[String]'
<code>log</code>	A resource may notify another resource to take action when its state changes. Specify a ' <u>resource[name]</u> ', the <u>:action</u> that resource should take, and then the <u>:timer</u> for that action. A resource may notify more than one resource; use a <u>notifies</u> statement for each resource to be notified.
<code>macports_package</code>	
<code>mdadm</code>	
<code>mount</code>	
<code>ohai</code>	A timer specifies the point during the chef-client run at which a notification is run. The following timers are available:
<code>openbsd_package</code>	<u>:before</u> Specifies that the action on a notified resource should be run before processing the resource block in which the notification is located.
<code>osx_profile</code>	<u>:delayed</u> Default. Specifies that a notification should be queued up, and then executed at the very end of the chef-client run.
<code>package</code>	<u>:immediate</u> , <u>:immediately</u> Specifies that a notification should be run immediately, per resource notified.
<code>pacman_package</code>	
<code>paludis_package</code>	
<code>perl</code>	
<code>portage_package</code>	
<code>powershell_script</code>	
<code>private_key</code>	The syntax for <u>notifies</u> is:

`public_key``notifies :action, 'resource[name]', :timer``python``path`**Ruby Type:** Array

An array of paths to use when searching for a command.

These paths are not added to the command's environment \$PATH. The default value uses the system path.

Warning

For example:

```
bash 'mycommand' do
  environment 'PATH' => "/my/path/to/bin:#{ENV['PA
end
```

`reboot``registry_key``remote_directory``remote_file``route``rpm_package``ruby``provider`**Ruby Type:** Chef Class

Optional. Explicitly specifies a provider. See “Providers” section below for more information.

`ruby_block``script``service``retries`**Ruby Type:** Integer

The number of times to catch exceptions and retry the resource. Default value: `0`.

`smartos_package``solaris_package``subversion``retry_delay`**Ruby Type:** Integer

The retry delay (in seconds). Default value: `2`.

`systemd_unit``template``user``returns`**Ruby Types:** Integer, Array

The return value for a command. This may be an array of accepted values. An exception is raised when the return value(s) do not match. Default value: `0`.

`windows_package``windows_service``subscribes`**Ruby Type:** Symbol, ‘Chef::Resource[String]’

A resource may listen to another resource, and then take action if the state of the resource being listened to changes. Specify a `'resource[name]'`, the `:action` to be taken, and then the `:timer` for that action.












`yum_package``yum_repository`

A timer specifies the point during the chef-client run at which a notification is run. The following timers are available:

`:before`

Specifies that the action on a notified resource should be run before processing the resource block in which the notification is located.

Examples (by Resource)**Templates****Cookbook Repo****metadata.rb****Cookbook Versions****Ruby**

	Chef DK	▼
	Managing the Server	▼
	Habitat	▼
	InSpec	▼
	Chef Automate	▼
	Extension APIs	▼
	Available on GitHub	
	Get Chef	
	Send Feedback	
	Support	
	Site Map	

:delayed

Default. Specifies that a notification should be queued up, and then executed at the very end of the chef-client run.

:immediate, :immediately

Specifies that a notification should be run immediately, per resource notified.

The syntax for subscribes is:

```
subscribes :action, 'resource[name]', :timer
```

timeout

Ruby Types: Integer, Float

The amount of time (in seconds) a command is to wait before timing out. Default value: 3600.

user

Ruby Types: String, Integer

The user name or user ID that should be changed before running a command.

umask

Ruby Types: String, Integer

The file mode creation mask, or umask.

Guards ¶

A guard property can be used to evaluate the state of a node during the execution phase of the chef-client run. Based on the results of this evaluation, a guard property is then used to tell the chef-client if it should continue executing a resource. A guard property accepts either a string value or a Ruby block value:

- A string is executed as a shell command. If the command returns 0, the guard is applied. If the command returns any other value, then the guard property is not applied. String guards in a **powershell_script** run Windows PowerShell commands and may return true in addition to 0.
- A block is executed as Ruby code that must return either true or false. If the block returns true, the guard property is applied. If the block returns false, the guard property is not applied.

A guard property is useful for ensuring that a resource is idempotent by allowing that resource to test for the desired

state as it is being executed, and then if the desired state is present, for the chef-client to do nothing.

Attributes

The following properties can be used to define a guard that is evaluated during the execution phase of the chef-client run:

not_if

Prevent a resource from executing when the condition returns true.

only_if

Allow a resource to execute only if the condition returns true.

Arguments

The following arguments can be used with the not_if or only_if guard properties:

:user

Specify the user that a command will run as. For example:

```
not_if 'grep adam /etc/passwd', :user => 'adam'
```

:group

Specify the group that a command will run as. For example:

```
not_if 'grep adam /etc/passwd', :group => 'adam'
```

:environment

Specify a Hash of environment variables to be set. For example:

```
not_if 'grep adam /etc/passwd', :environment =>
  'HOME' => '/home/adam'
}
```

:cwd

Set the current working directory before running a command. For example:

```
not_if 'grep adam passwd', :cwd => '/etc'
```

:timeout

Set a timeout for a command. For example:

```
not_if 'sleep 10000', :timeout => 10
```

Providers ¶

Where a resource represents a piece of the system (and its

desired state), a provider defines the steps that are needed to bring that piece of the system from its current state into the desired state.

The chef-client will determine the correct provider based on configuration data collected by Ohai at the start of the chef-client run. This configuration data is then mapped to a platform and an associated list of providers.

Generally, it's best to let the chef-client choose the provider, and this is (by far) the most common approach. However, in some cases, specifying a provider may be desirable. There are two approaches:

- Use a more specific short name—`yum_package "foo" do` instead of `package "foo" do`, `script "foo" do` instead of `bash "foo" do`, and so on—when available
- Use the `provider` property within the resource block to specify the long name of the provider as a property of a resource. For example: `provider Chef::Provider::Long::Name`

This resource has the following providers:

`Chef::Provider::Script`, `script`

When this short name is used, the chef-client will determine the correct provider during the chef-client run.

`Chef::Provider::Script::Bash`, `bash`

The provider for the Bash command interpreter.

Examples ¶

The following examples demonstrate various approaches for using resources in recipes. If you want to see examples of how Chef uses resources in recipes, take a closer look at the cookbooks that Chef authors and maintains: <https://github.com/chef-cookbooks>.

Use a named provider to run a script

```
bash 'install_something' do
  user 'root'
  cwd '/tmp'
  code <<-EOH
  wget http://www.example.com/tarball.tar.gz
  tar -zxf tarball.tar.gz
  cd tarball
  ./configure
  make
  make install
  EOH
end
```

Install a file from a remote location using bash

The following is an example of how to install the `foo123` module for Nginx. This module adds shell-style functionality to an Nginx configuration file and does the following:

- Declares three variables
- Gets the Nginx file from a remote location
- Installs the file using Bash to the path specified by the `src_filepath` variable

*# the following code sample is similar to the ``uplo
recipe in the ``nginx`` cookbook:
<https://github.com/chef-cookbooks/nginx>*

```
src_filename = "foo123-nginx-module-v#{
  node['nginx']['foo123']['version']
}.tar.gz"
src_filepath = "#{Chef::Config['file_cache_path']}/#
extract_path = "#{
  Chef::Config['file_cache_path']
}/nginx_foo123_module/#{
  node['nginx']['foo123']['checksum']
}"

remote_file 'src_filepath' do
  source node['nginx']['foo123']['url']
  checksum node['nginx']['foo123']['checksum']
  owner 'root'
  group 'root'
  mode '0755'
end

bash 'extract_module' do
  cwd ::File.dirname(src_filepath)
  code <<-EOH
    mkdir -p #{extract_path}
    tar xzf #{src_filename} -C #{extract_path}
    mv #{extract_path}/*/* #{extract_path}/
  EOH
  not_if { ::File.exist?(extract_path) }
end
```

Install an application from git using bash

The following example shows how Bash can be used to install a plug-in for `rbenv` named `ruby-build`, which is located in git version source control. First, the application is synchronized, and then Bash changes its working directory to the location in which `ruby-build` is located, and then runs a command.

```
git "#{Chef::Config[:file_cache_path]}/ruby-build"
  repository 'git://github.com/sstephenson/ruby-bui
  reference 'master'
  action :sync
end

bash 'install_ruby_build' do
  cwd "#{Chef::Config[:file_cache_path]}/ruby-build
  user 'rbenv'
```



```

group 'rbenv'
code <<-EOH
  ./install.sh
EOH
environment 'PREFIX' => '/usr/local'
end

```

To read more about [ruby-build](https://github.com/sstephenson/ruby-build), see here: <https://github.com/sstephenson/ruby-build>.

Store certain settings

The following recipe shows how an attributes file can be used to store certain settings. An attributes file is located in the `attributes/` directory in the same cookbook as the recipe which calls the attributes file. In this example, the attributes file specifies certain settings for Python that are then used across all nodes against which this recipe will run.

Python packages have versions, installation directories, URLs, and checksum files. An attributes file that exists to support this type of recipe would include settings like the following:

```

default['python']['version'] = '2.7.1'

if python['install_method'] == 'package'
  default['python']['prefix_dir'] = '/usr'
else
  default['python']['prefix_dir'] = '/usr/local'
end

default['python']['url'] = 'http://www.python.org/ft
default['python']['checksum'] = '80e387...85fd61'

```

and then the methods in the recipe may refer to these values. A recipe that is used to install Python will need to do the following:

- Identify each package to be installed (implied in this example, not shown)
- Define variables for the package `version` and the `install_path`
- Get the package from a remote location, but only if the package does not already exist on the target system
- Use the **bash** resource to install the package on the node, but only when the package is not already installed

the following code sample comes from the ``oc-ngi

```

version = node['python']['version']
install_path = "#{node['python']['prefix_dir']}/lib/

remote_file "#{Chef::Config[:file_cache_path]}/Pytho
  source "#{node['python']['url']}/#{version}/Python
  checksum node['python']['checksum']
  mode '0755'
  not_if { ::File.exist?(install_path) }
end

```

```
bash 'build-and-install-python' do
  cwd Chef::Config[:file_cache_path]
  code <<-EOF
    tar -jxvf Python-#{version}.tar.bz2
    (cd Python-#{version} && ./configure #{configure}
    (cd Python-#{version} && make && make install)
  EOF
  not_if { ::File.exist?(install_path) }
end
```

© Copyright: This work is licensed under a Creative Commons Attribution 3.0 Unported License. This page is about: current version of Chef. Provide feedback on Chef documentation.