COOKBOOK

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

Syntax

A **file** resource block manages files that exist on nodes. For example, to write the home page for an Apache website:

```
file '/var/www/customers/public_html/index.php' do
content '<html>This is a placeholder for the home page.</html>'
mode '0755'
owner 'web_admin'
group 'web_admin'
end
```

where

- '/var/www/customers/public_html/index.php' is path to the file and also the filename to be managed
- content defines the contents of the file

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

```
file 'name' do
atomic_update              True, False
backup                  False, Integer
checksum                  String
content                 String
force_unlink              True, False
group                  String, Integer
inherits               True, False
manage_symlink_source      True, False
mode                    String, Integer
notifies               # see description
owner                   String, Integer
path                  String # defaults to 'name' if not specified
rights                Hash
sensitive               True, False
subscribes                # see description
```

```
verify                    String, Block
action                     Symbol # defaults to :create if not specified
end
```

where

- file is the resource
- name is the name of the resource block; when the path property is not specified as part of a recipe, name is also the path to the file
- content specifies the contents of the file
- action identifies the steps the chef-client will take to bring the node into the desired state
- atomic_update, backup, checksum, content, force_unlink, group, inherits, manage_symlink_source, mode, owner, path, rights, sensitive, and verify are properties of this resource, with the Ruby type shown.

Actions¶

This resource has the following actions:

:create

Default. Create a file. If a file already exists (but does not match), update that file to match.

:create_if_missing

Create a file only if the file does not exist. When the file exists, nothing happens.

:delete

Delete a file.

:nothing

Define this resource block to do nothing until notified by another resource to take action. When this resource is notified, this resource block is either run immediately or it is queued up to be run at the end of the Chef Client run.

:touch

Touch a file. This updates the access (atime) and file modification (mtime) times for a file.

Properties¶

This resource has the following properties:

atomic_update

Ruby Types: True, False
Perform atomic file updates on a per-resource basis. Set to true for atomic file updates. Set to false for non-atomic file updates. This setting overrides file_atomic_update, which is a global setting found in the client.rb file. Default value: true.

backup

Ruby Types: False, Integer
The number of backups to be kept in /var/chef/backup (for UNIX- and Linux-based platforms) or C:/chef/backup (for the Microsoft Windows platform). Set to false to prevent backups from being kept. Default value: 5.

checksum

Ruby Types: String
The SHA-256 checksum of the file. Use to ensure that a specific file is used. If the checksum does not match, the file is not used. Default value: no checksum required.

content

Ruby Type: String
A string that is written to the file. The contents of this property replace any previous content when this property has something other than the default value. The default behavior will not modify content.

force_unlink

Ruby Types: True, False
How the chef-client handles certain situations when the target file turns out not to be a file. For example, when a target file is actually a symlink. Set to true for the chef-client delete the non-file target and replace it with the specified file. Set to false for the chef-client to raise an error. Default value: false.

group

Ruby Types: Integer, String

A string or ID that identifies the group owner by group name, including fully qualified group names such as domain\group or group@domain. If this value is not specified, existing groups remain unchanged and new group assignments use the default POSIX group (if available).

ignore_failure

Ruby Types: True, False
Continue running a recipe if a resource fails for any reason. Default value: false.

inherits

Ruby Types: True, False
Microsoft Windows only. Whether a file inherits rights from its parent directory. Default value: true.

manage_symlink_source

Ruby Types: True, False | **Default Value:** true (with warning)
Change the behavior of the file resource if it is pointed at a symlink. When this value is set to true, the Chef client will manage the symlink's permissions or will replace the symlink with a normal file if the resource has content. When this value is set to false, Chef will follow the symlink and will manage the permissions and content of symlink's target file.
The default behavior is true but emits a warning that the default value will be changed to false in a future version; setting this explicitly to true or false suppresses this warning.

Execute:

Syntax

An **execute** resource block typically executes a single command that is unique to the environment in which a recipe will run. Some **execute** resource commands are run by themselves, but often they are run in combination with other Chef resources. For example, a single command that is run by itself:

```
execute 'apache_configtest' do
command '/usr/sbin/apachectl configtest'
end
```

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

```
execute 'name' do
command                String, Array # defaults to 'name' if not specified
creates               String
cwd                  String
environment              Hash # env is an alias for environment
group                String, Integer
live_stream              True, False
notifies              # see description
returns               Integer, Array
sensitive             True, False
subscribes              # see description
timeout               Integer, Float
umask                 String, Integer
user                 String
password               String
domain                String
action               Symbol # defaults to :run if not specified
end
```

where

- execute is the resource
- name is the name of the resource block
- command is the command to be run
- action identifies the steps the chef-client will take to bring the node into the desired state
- command, creates, cwd, environment, group, live_stream, returns, sensitive, timeout, user, password, domain 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 command.

Properties¶

This resource has the following properties:

`command`

Ruby Types: String, Array
The name of the command to be executed. Default value: the `name` of the resource block See “Syntax” section above for more information.

Note

Use the **execute** resource to run a single command. Use multiple **execute** resource blocks to run multiple commands.

`creates`

Ruby Type: String
Prevent a command from creating a file when that file already exists.

`cwd`

Ruby Type: String
The current working directory from which a command is run.

`environment`

Ruby Type: Hash
A Hash of environment variables in the form of `({"ENV_VARIABLE" => "VALUE"})`. (These variables must exist for a command to be run successfully.)

`group`

Ruby Types: String, Integer
The group name or group ID that must be changed before running a command.

`ignore_failure`

Ruby Types: True, False
Continue running a recipe if a resource fails for any reason. Default value: `false`.

`live_stream`

Ruby Types: True, False
Send the output of the command run by this **execute** resource block to the chef-client event stream. Default value: `false`.
New in Chef Client 12.6.

`notifies`

Ruby Type: Symbol, 'Chef::Resource[String]'
A resource may notify another resource to take action when its state changes. Specify a `'resource[name]'`, the `:action` that resource should take, and then the `:timer` for that action. A resource may notify more than one resource; use a `notifies` statement for each resource to be notified.
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.

`: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 `notifies` is:

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

`retries`

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

`retry_delay`

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

`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.

sensitive

Ruby Types: True, False
Ensure that sensitive resource data is not logged by the chef-client. Default value: false.

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.
Note that subscribes does not apply the specified action to the resource that it listens to - for example:

```
file '/etc/nginx/ssl/example.crt' do
mode '0600'
owner 'root'
end

service 'nginx' do
subscribes :reload, 'file[/etc/nginx/ssl/example.crt]', :immediately
end
```

In this case the subscribes property reloads the nginx service whenever its certificate file, located under /etc/nginx/ssl/example.crt, is updated. subscribes does not make any changes to the certificate file itself, it merely listens for a change to the file, and executes the :reload action for its resource (in this example nginx) when a change is detected.
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.

: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
The user name of the user identity with which to launch the new process. Default value: *nil*. The user name may optionally be specifed with a domain, i.e. *domainuser* or *user@my.dns.domain.com* via Universal Principal Name (UPN)format. It can also be specified without a domain simply as user if the domain is instead specified using the *domain* attribute. On Windows only, if this property is specified, the *password* property must be specified.

`password`

Ruby Types: String
Windows only: The password of the user specified by the *user* property. Default value: *nil*. This property is mandatory if *user* is specified on Windows and may only be specified if *user* is specified. The *sensitive*property for this resource will automatically be set to true if password is specified.

`domain`

Ruby Types: String
Windows only: The domain of the user user specified by the *user* property. Default value: *nil*. If not specified, the user name and password specified by the *user* and *password* properties will be used to resolve that user against the domain in which the system running Chef client is joined, or if that system is not joined to a domain it will resolve the user as a local account on

that system. An alternative way to specify the domain is to leave this property unspecified and specify the domain as part of the *user* property.

umask

Ruby Types: String, Integer
The file mode creation mask, or umask.

Logging:

Use the **log** resource to create log entries

Syntax:
A **log** resource block adds messages to the log file based on events that occur during the Chef Client run:

```
log 'message' do
message 'A message add to the log.'
level :info
end
```

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

```
log 'name' do
level                    Symbol
message                     String # defaults to 'name' if not specified
notifies                  # see description
subscribes                   # see description
action                    Symbol # defaults to :write if not specified
end
```

where

- log is the resource
- name is the name of the resource block
- message is the log message to write

- action identifies the steps the Chef Client will take to bring the node into the desired state
- level and message 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

Define this resource block to do nothing until notified by another resource to take action. When this resource is notified, this resource block is either run immediately or it is queued up to be run at the end of the Chef Client run.

:write

Default. Write to log.

Properties¶

This resource has the following properties:

ignore_failure

Ruby Types: True, False | **Default Value:** false
Continue running a recipe if a resource fails for any reason.

level

Ruby Type: Symbol | **Default Value:** :info
The level of logging that is to be displayed by the Chef Client. Options (in order of priority): :debug, :info, :warn, :error, and :fatal.

message

Ruby Type: String
The message to be added to a log file. Default value: the name of the resource block See “Syntax” section above for more information.

notifies

Ruby Type: Symbol, ‘Chef::Resource[String]’
A resource may notify another resource to take action when its state changes. Specify a 'resource[name]', the :action that resource should take,

and then the :timer for that action. A resource may notify more than one resource; use a notifies statement for each resource to be notified.
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.

: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 notifies is:

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

retries

Ruby Type: Integer | **Default Value:** 0
The number of times to catch exceptions and retry the resource.

retry_delay

Ruby Type: Integer | **Default Value:** 2
The retry delay (in seconds).

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.
Note that subscribes does not apply the specified action to the resource that it listens to - for example:

```
file '/etc/nginx/ssl/example.crt' do
mode '0600'
owner 'root'
```

```
end

service 'nginx' do
subscribes :reload, 'file[/etc/nginx/ssl/example.crt]', :immediately
end
```

In this case the subscribes property reloads the nginx service whenever its certificate file, located under /etc/nginx/ssl/example.crt, is updated. subscribes does not make any changes to the certificate file itself, it merely listens for a change to the file, and executes the :reload action for its resource (in this example nginx) when a change is detected.
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.

: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
```

Set debug logging level

```
log 'a debug string' do
level :debug
end
```

Use the **service** resource to manage a service.

Syntax¶

A **service** resource block manages the state of a service. For example:

```
service "tomcat" do
action :start
end
```

will start the Apache Tomcat service.

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

```
service 'name' do
init_command              String
notifies              # see description
options               Array, String
pattern               String
priority              Integer, String, Hash
reload_command            String
restart_command           String
service_name            String # defaults to 'name' if not specified
start_command             String
status_command            String
stop_command              String
subscribes              # see description
supports                Hash
timeout                 Integer # Microsoft Windows only
action                Symbol # defaults to :nothing if not specified
end
```

where

- service is the resource
- name is the name of the resource block; when the path property is not specified, name is also the path to the directory, from the root
- action identifies the steps the chef-client will take to bring the node into the desired state
- init_command, options, pattern, priority, reload_command, restart_command, service_name, start_command, status_command, stop_command, supports, and timeout 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:

:disable

Disable a service. This action is equivalent to a Disabled startup type on the Microsoft Windows platform. This action is not supported when using System Resource Controller (SRC) on the AIX platform because System Resource Controller (SRC) does not have a standard mechanism for enabling and disabling services on system boot.

:enable

Enable a service at boot. This action is equivalent to an Automatic startup type on the Microsoft Windows platform. This action is not supported when using System Resource Controller (SRC) on the AIX platform because System Resource Controller (SRC) does not have a standard mechanism for enabling and disabling services on system boot.

:nothing

Default. Do nothing with a service.

:reload

Reload the configuration for this service.

:restart

Restart a service.

:start

Start a service, and keep it running until stopped or disabled.

:stop

Stop a service.

Properties¶

This resource has the following properties:

ignore_failure

Ruby Types: True, False

Continue running a recipe if a resource fails for any reason. Default value: false.

init_command

Ruby Type: String

The path to the init script that is associated with the service. Use init_command to prevent the need to specify overrides for the start_command, stop_command, and restart_command properties. When this property is not specified, the chef-client will use the default init command for the service provider being used.

notifies

Ruby Type: Symbol, ‘Chef::Resource[String]’
A resource may notify another resource to take action when its state changes. Specify a 'resource[name]', the :action that resource should take, and then the :timer for that action. A resource may notify more than one resource; use a notifies statement for each resource to be notified.
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.

: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 notifies is:

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

options

Ruby Type: Array, String
Solaris platform only. Options to pass to the service command. See the svcadm manual for details of possible options.

pattern

Ruby Type: String
The pattern to look for in the process table. Default value: service_name.

priority

Ruby Types: Integer, String, Hash
Debian platform only. The relative priority of the program for start and shutdown ordering. May be an integer or a Hash. An integer is used to define the start run levels; stop run levels are then 100-integer. A Hash is used to define values for specific run levels. For example, `{ 2 => [:start, 20], 3 => [:stop, 55] }` will set a priority of twenty for run level two and a priority of fifty-five for run level three.

`reload_command`

Ruby Type: String
The command used to tell a service to reload its configuration.

`restart_command`

Ruby Type: String
The command used to restart a service.

`retries`

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

`retry_delay`

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

`service_name`

Ruby Type: String
The name of the service. Default value: the `name` of the resource block See "Syntax" section above for more information.

`start_command`

Ruby Type: String
The command used to start a service.

`status_command`

Ruby Type: String
The command used to check the run status for a service.

`stop_command`

Ruby Type: String

The command used to stop a service.

Reboot:

Use the **reboot** resource to reboot a node, a necessary step with some installations on certain platforms. This resource is supported for use on the Microsoft Windows, macOS, and Linux platforms. New in Chef Client 12.0.
New in Chef Client 12.0

Syntax¶

A **reboot** resource block reboots a node:

```
reboot 'app_requires_reboot' do
action :request_reboot
reason 'Need to reboot when the run completes successfully.'
delay_mins 5
end
```

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

```
reboot 'name' do
delay_mins                Fixnum
notifies               # see description
reason                  String
subscribes               # see description
action                 Symbol
end
```

where

- `reboot` is the resource
- `name` is the name of the resource block
- `action` identifies the steps the chef-client will take to bring the node into the desired state
- `delay_mins` and `reason` 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:

`:cancel`

Cancel a reboot request.

:nothing

Define this resource block to do nothing until notified by another resource to take action. When this resource is notified, this resource block is either run immediately or it is queued up to be run at the end of the Chef Client run.

:reboot_now

Reboot a node so that the chef-client may continue the installation process.

:request_reboot

Reboot a node at the end of a chef-client run.

Properties¶

This resource has the following properties:

delay_mins

Ruby Type: Fixnum

The amount of time (in minutes) to delay a reboot request.

ignore_failure

Ruby Types: True, False

Continue running a recipe if a resource fails for any reason. Default value: false.

notifies

Ruby Type: Symbol, 'Chef::Resource[String]'

A resource may notify another resource to take action when its state changes. Specify a 'resource[name]', the :action that resource should take, and then the :timer for that action. A resource may notify more than one resource; use a notifies statement for each resource to be notified.

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

:immediate, :immediately

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

reason

Ruby Type: String

A string that describes the reboot action.

retries

Ruby Type: Integer

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

retry_delay

Ruby Type: Integer

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

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.

Note that subscribes does not apply the specified action to the resource that it listens to - for example:

```
file '/etc/nginx/ssl/example.crt' do
mode '0600'
owner 'root'
end

service 'nginx' do
subscribes :reload, 'file[/etc/nginx/ssl/example.crt]', :immediately
end
```