Overview

This is a Puppet package provider for Chocolatey, which is like apt-get, but for Windows. Check the module's metadata.json for compatible Puppet and Puppet Enterprise versions.

Module Description

This is the official module for working with the Chocolatey package manager.

This module supports all editions of Chocolatey, including FOSS, Professional and Chocolatey for Business.

This module is able to:

- Install Chocolatey
- Work with custom location installations
- Configure Chocolatey
- Use Chocolatey as a package provider

Why Chocolatey

Chocolatey closely mimics how package managers on other operating systems work. If you can imagine the built-in provider for Windows versus Chocolatey, take a look at the use case of installing git:

```
# Using built-in provider
package { "Git version 1.8.4-preview20130916":
  ensure     => installed,
  source     => 'C:\temp\Git-1.8.4-preview20130916.exe',
  install_options => ['/VERYSILENT']
}
```

```
# Using Chocolatey (set as default for Windows)
package { 'git':
  ensure    => latest,
}
```

With the built-in provider:

- The package name must match ***exactly*** the name from installed programs.
- The package name has issues with unicode characters.
- The source must point to the location of the executable installer.
- It cannot `ensure => latest`. Read about handling versions and upgrades in the Puppet documentation.

With Chocolatey's provider:

- The package name only has to match the name of the package, which can be whatever you choose.
- The package knows how to install the software silently.
- The package knows where to get the executable installer.
- The source is able to specify different Chocolatey feeds.
- Chocolatey makes `package` more platform agnostic, because it looks exactly like other platforms.

For reference, read about the provider features available from the built-in provider, compared to other package managers:

Provider	holdable	install options	installable	package settings	purgeable	reinstallable	uninstall options	uninstallab
Windows		x	x				x	x
Chocolatey	x	x	x				x	x
apt	x	x	x		x			x
yum		x	x		x			x

Setup

What Chocolatey affects

Chocolatey affects your system and what software is installed on it, ranging from tools and portable software, to natively installed applications.

Setup Requirements

Chocolatey requires the following components:

- Powershell v2+ (Installed on most systems by default)
- .NET Framework v4+

Beginning with Chocolatey provider

Install this module via any of these approaches:

- Puppet Forge
- git-submodule (tutorial)
- librarian-puppet
- r10k

Usage

Manage Chocolatey installation

Ensure Chocolatey is installed and configured:

```
include chocolatey
```

Override default Chocolatey install location

```
class {'chocolatey':
  choco_install_location => 'D:\secured\choco',
}
```

NOTE: This will affect suitability on first install. There are also special considerations for `C:\Chocolatey` as an install location, see `choco_install_location` for details.

Use an internal chocolatey.nupkg for Chocolatey installation

```
class {'chocolatey':
  chocolatey_download_url         => 'https://internalurl/to/chocolatey.nupkg',
  use_7zip                        => false,
  choco_install_timeout_seconds   => 2700,
}
```

Use a file chocolatey.0.9.9.9.nupkg for installation

```
class {'chocolatey':
  chocolatey_download_url         => 'file:///c:/location/of/chocolatey.0.9.9.9.nupkg',
  use_7zip                        => false,
  choco_install_timeout_seconds   => 2700,
}
```

Specify the version of chocolatey by class parameters

```
class {'chocolatey':
  chocolatey_download_url         => 'file:///c:/location/of/chocolatey.0.9.9.9.nupkg',
  use_7zip                        => false,
  choco_install_timeout_seconds   => 2700,
  chocolatey_version              => '0.9.9.9',
}
```

Log chocolatey bootstrap installer script output

```
class {'chocolatey':
  log_output              => true,
}
```

Configuration

If you have Chocolatey 0.9.9.x or above, you can take advantage of configuring different aspects of Chocolatey.

Sources Configuration

You can specify sources that Chocolatey uses by default, along with priority.

Requires Chocolatey v0.9.9.0+.

Disable the default community repository source

```
chocolateysource {'chocolatey':
  ensure => disabled,
}
```

Set a priority on a source

```
chocolateysource {'chocolatey':
  ensure   => present,
  location => 'https://chocolatey.org/api/v2',
  priority => 1,
}
```

Add credentials to a source

```
chocolateysource {'sourcename':
  ensure   => present,
  location => 'https://internal/source',
  user     => 'username',
  password => 'password',
}
```

NOTE: Chocolatey encrypts the password in a way that is not verifiable. If you need to rotate passwords, you cannot use this resource to do so unless you also change the location, user, or priority (because those are ensurable properties).

Features Configuration

You can configure features that Chocolatey has available. Run `choco feature list` to see the available configuration features.

Requires Chocolatey v0.9.9.0+.

Enable Auto Uninstaller

Uninstall from Programs and Features without requiring an explicit uninstall script.

```
chocolateyfeature {'autouninstaller':
  ensure => enabled,
}
```

Disable Use Package Exit Codes

Requires 0.9.10+ for this feature.

Use Package Exit Codes - Allows package scripts to provide exit codes. With this enabled, Chocolatey uses package exit codes for exit when non-zero (this value can come from a dependency package). Chocolatey defines valid exit codes as 0, 1605, 1614, 1641, 3010. With this feature disabled, Chocolatey exits with a 0 or a 1 (matching previous behavior).

```
chocolateyfeature {'usepackageexitcodes':
  ensure => disabled,
}
```

Enable Virus Check

Requires 0.9.10+ and Chocolatey Pro / Business for this feature.

Virus Check - Performs virus checking on downloaded files. *(Licensed versions only.)*

```
chocolateyfeature {'viruscheck':
  ensure => enabled,
}
```

Enable FIPS Compliant Checksums

Requires 0.9.10+ for this feature.

Use FIPS Compliant Checksums - Ensures checksumming done by Chocolatey uses FIPS compliant algorithms. *Not recommended unless required by FIPS Mode.* Enabling on an existing installation could have unintended consequences related to upgrades or uninstalls.

```
chocolateyfeature {'usefipscompliantchecksums':
  ensure => enabled,
}
```

Config configuration

You can configure config values that Chocolatey has available. Run `choco config list` to see the config settings available (just the config settings section).

Requires Chocolatey v0.9.10.0+.

Set cache location

The cache location defaults to the TEMP directory. You can set an explicit directory to cache downloads to instead of the default.

```
chocolateyconfig {'cachelocation':
  value  => "c:\\downloads",
}
```

Unset cache location

Removes cache location setting, returning the setting to its default.

```
chocolateyconfig {'cachelocation':
  ensure => absent,
}
```

Use an explicit proxy

When using Chocolatey behind a proxy, set `proxy` and optionally `proxyUser` and `proxyPassword`.

NOTE: The `proxyPassword` value is not verifiable.

```
chocolateyconfig {'proxy':
  value  => 'https://someproxy.com',
}

chocolateyconfig {'proxyUser':
  value  => 'bob',
}

# not verifiable
chocolateyconfig {'proxyPassword':
  value  => 'securepassword',
}
```

Set Chocolatey as Default Windows Provider

If you want to set this provider as the site-wide default, add to your `site.pp`:

```
if $::kernel == 'windows' {
  Package { provider => chocolatey, }
}

# OR

case $operatingsystem {
  'windows': {
    Package { provider => chocolatey, }
  }
}
```

Packages

With all options

```
package { 'notepadplusplus':
  ensure            => installed|latest|'1.0.0'|absent,
  provider          => 'chocolatey',
  install_options   => ['-pre','-params','"','param1','param2','"'],
  uninstall_options => ['-r'],
  source            => 'https://myfeed.example.com/api/v2',
}
```

- Supports `installable` and `uninstallable`.
- Supports `versionable` so that `ensure => '1.0'` works.
- Supports `upgradeable`.
- Supports `latest` (checks upstream), `absent` (uninstall).
- Supports `install_options` for pre-release, and other command-line options.
- Supports `uninstall_options` for pre-release, and other command-line options.
- Supports `holdable`, requires Chocolatey v0.9.9.0+.

Simple install

```
package { 'notepadplusplus':
  ensure   => installed,
  provider => 'chocolatey',
}
```

To always ensure using the newest version available

```
package { 'notepadplusplus':
  ensure   => latest,
  provider => 'chocolatey',
}
```

To ensure a specific version

```
package { 'notepadplusplus':
  ensure   => '6.7.5',
  provider => 'chocolatey',
}
```

To specify custom source

```
package { 'notepadplusplus':
  ensure   => '6.7.5',
  provider => 'chocolatey',
  source   => 'C:\local\folder\packages',
}
```

```
package { 'notepadplusplus':
  ensure   => '6.7.5',
  provider => 'chocolatey',
  source   => '\\unc\source\packages',
}
```

```
package { 'notepadplusplus':
  ensure   => '6.7.5',
  provider => 'chocolatey',
  source   => 'https://custom.nuget.odata.feed/api/v2/',
}
```

```
package { 'notepadplusplus':
  ensure   => '6.7.5',
  provider => 'chocolatey',
  source   => 'C:\local\folder\packages;https://chocolatey.org/api/v2/',
}
```

Install options with spaces

Spaces in arguments **must always** be covered with a separation. Shown below is an example of how you configure `-installArgs "/VERYSILENT /NORESTART"`.

```
package {'launchy':
  ensure          => installed,
  provider        => 'chocolatey',
  install_options => ['-override', '-installArgs', '"', '/VERYSILENT', '/NORESTART', '"'],
}
```

Install options with quotes or spaces

The underlying installer may need quotes passed to it. This is possible, but not as intuitive. The example below covers passing `/INSTALLDIR="C:\Program Files\somewhere"`.

For this to be passed through with Chocolatey, you need a set of double quotes surrounding the argument and two sets of double quotes surrounding the item that must be quoted (see how to pass/options/switches). This makes the string look like `-installArgs "/INSTALLDIR=""C:\Program Files\somewhere"""` for proper use with Chocolatey.

Then, for Puppet to handle that appropriately, you must split on ***every*** space. Yes, on **every** space you must split the string or the result comes out incorrectly. This means it will look like the following:

```
install_options => ['-installArgs',
  '"/INSTALLDIR=""C:\Program', 'Files\somewhere"""']
```

Make sure you have all of the right quotes - start it off with a single double quote, then two double quotes, then close it all by closing the two double quotes and then the single double quote or a possible three double quotes at the end.

```
package {'mysql':
  ensure          => latest,
  provider        => 'chocolatey',
  install_options => ['-override', '-installArgs',
    '"/INSTALLDIR=""C:\Program', 'Files\somewhere"""'],
}
```

You can split it up a bit for readability if it suits you:

```
package {'mysql':
  ensure          => latest,
  provider        => 'chocolatey',
  install_options => ['-override', '-installArgs', '"'
    '/INSTALLDIR=""C:\Program', 'Files\somewhere""',
    '"'],
}
```

Note: The above is for Chocolatey v0.9.9+. You may need to look for an alternative method to pass args if you have 0.9.8.x and below.

Reference

Classes

Public classes

- `chocolatey`

Private classes

- `chocolatey::install.pp`: Ensures Chocolatey is installed.
- `chocolatey::config.pp`: Ensures Chocolatey is configured.

Facts

- `chocolateyversion` - The version of the installed Chocolatey client (could also be provided by class parameter `chocolatey_version`).
- `choco_install_path` - The location of the installed Chocolatey client (could also be provided by class parameter `choco_install_location`).

Types/Providers

- Chocolatey provider
- Chocolatey source configuration
- Chocolatey feature configuration

Package provider: Chocolatey

Chocolatey implements a package type with a resource provider, which is built into Puppet.

This provider supports the `install_options` and `uninstall_options` attributes, which allow command-line options to be passed to the `choco` command. These options should be specified as documented below.

- Required binaries: `choco.exe`, usually found in `C:\Program Data\chocolatey\bin\choco.exe`.
  - The binary is searched for using the environment variable `ChocolateyInstall`, then by two known locations (`C:\Chocolatey\bin\choco.exe` and `C:\ProgramData\chocolatey\bin\choco.exe`).

- Supported features: `install_options`, `installable`, `uninstall_options`, `uninstallable`, `upgradeable`, `versionable`.

NOTE: the root of `C:\` is not a secure location by default, so you may want to update the security on the folder.

Properties/Parameters

`ensure`

(**Property**: This attribute represents a concrete state on the target system.)

Specifies what state the package should be in. You can choose which package to retrieve by specifying a version number or `latest` as the ensure value. Valid options: `present` (also called `installed`), `absent`, `latest`, `held` or a version number. Default: `installed`.

`install_options`
Specifies an array of additional options to pass when installing a package. These options are package-specific, and should be documented by the software vendor. One commonly implemented option is `INSTALLDIR`:

```
package {'launchy':
  ensure          => installed,
  provider        => 'chocolatey',
  install_options => ['-installArgs', '"', '/VERYSILENT', '/NORESTART', '"'],
}
```

NOTE: The above method of single quotes in an array is the only method you should use in passing `install_options` with the Chocolatey provider. There are other ways to do it, but they are passed through to Chocolatey in ways that may not be sufficient.

This is the **only** place in Puppet where backslash separators should be used. Note that backslashes in double-quoted strings *must* be double-escaped and backslashes in single-quoted strings *may* be double-escaped.

`name`
Specifies the package name. This is the name that the packaging system uses internally. Valid options: String. Default: The resource's title.

`provider`
Required. Sets the specific backend to use for the `package` resource. Chocolatey is not the default provider for Windows, so it must be specified (or by using a resource default, shown in the Usage section above). Valid options: `'chocolatey'`.

`source`
Specifies where to find the package file. Use this to override the default source(s). Valid options: String of either an absolute path to a local directory containing packages stored on the target system, a URL (to OData feeds), or a network drive path. Chocolatey maintains default sources in its configuration file that it uses by default.

Puppet will not automatically retrieve source files for you, and usually just passes the value of the source to the package installation command. You can use a `file` resource if you need to manually copy package files to the target system.

`uninstall_options`
Specifies an array of additional options to pass when uninstalling a package. These options are package-specific, and should be documented by the software vendor.

```
package {'launchy':
  ensure            => absent,
  provider          => 'chocolatey',
  uninstall_options => ['-uninstallargs', '"', '/VERYSILENT', '/NORESTART', '"'],
}
```

The above method of single quotes in an array is the only method you should use in passing `uninstall_options` with the Chocolatey provider. There are other ways to do it, but they are passed to Chocolatey in ways that may not be sufficient.

NOTE: This is the **only** place in Puppet where backslash separators should be used. Backslashes in double-quoted strings *must* be double-escaped and backslashes in single-quoted strings *may* be double-escaped.

ChocolateySource

Allows managing default sources for Chocolatey. A source can be a folder, a CIFS share, a NuGet Http OData feed, or a full Package Gallery. Learn more about sources at How To Host Feed. Requires Chocolatey v0.9.9.0+.

Properties/Parameters

`name`

Specifies the name of the source. Used for uniqueness. Also sets the `location` to this value if `location` is unset. Valid options: String. Default: The resource's title.

`ensure`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies what state the source should be in. Default: `present`. Valid options: `present`, `disabled`, or `absent`. `disabled` should only be used with existing sources.

`location`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies the location of the source repository. Valid options: String of a URL pointing to an OData feed (such as chocolatey/chocolatey_server), a CIFS (UNC) share, or a local folder. Required when `ensure => present` (`present` is default value for `ensure`).

`user`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies an optional user name for authenticated feeds. Requires at least Chocolatey v0.9.9.0. Default: `nil`. Specifying an empty value is the same as setting the value to `nil` or not specifying the property at all.

`password`
Specifies an optional user password for authenticated feeds. Not ensurable. Value cannot be checked with current value. If you need to update the password, update another setting as well to force the update. Requires at least Chocolatey v0.9.9.0. Default: `nil`. Specifying an empty value is the same as setting the value to `nil` or not specifying the property at all.

`priority`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies an optional priority for explicit feed order when searching for packages across multiple feeds. The lower the number, the higher the priority. Sources with a 0 priority are considered no priority and are added after other sources with a priority number. Requires at least Chocolatey v0.9.9.9. Default: `0`.

ChocolateyFeature

Allows managing features for Chocolatey. Features are configurations that act as switches to turn on or off certain aspects of how Chocolatey works. Learn more about features in the Chocolatey documentation. Requires Chocolatey v0.9.9.0+.

Properties/Parameters

`name`
Specifies the name of the feature. Used for uniqueness. Valid options: String. Default: The resource's title.

`ensure`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies what state the feature should be in. Valid options: `enabled` or `disabled`. Default: `disabled`.

ChocolateyConfig

Allows managing config settings for Chocolatey. Configuration values provide settings for users to configure aspects of Chocolatey and the way it functions. Similar to features, except allow for user configured values. Learn more about config settings at Config. Requires Chocolatey v0.9.9.9+.

Properties/Parameters

`name`
(**Namevar**: If ommitted, this parameter's value will default to the resource's title.)

Specifies the name of the config setting. Used for uniqueness. Puppet is not able to easily manage any values that include "password" in the key name because they will be encrypted in the configuration file.

`ensure`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies what state the config should be in. Valid options: `present` or `absent`. Default: `present`.

`value`
(**Property**: This parameter represents a concrete state on the target system.)

Specifies the value of the config setting. If the name includes "password", then the value is not ensurable due to being encrypted in the configuration file.

Class: chocolatey

Manages installation and configuration of Chocolatey itself.

Parameters

`choco_install_location`
Specifies where Chocolatey install should be located. Valid options: Must be an absolute path starting with a drive letter, for example: `c:\`. Default: The currently detected install location based on the `ChocolateyInstall` environment variable. If not specified, falls back to `'C:\ProgramData\chocolatey'`.

NOTE: Puppet can install Chocolatey and configure Chocolatey install packages during the same run *UNLESS* you specify this setting. This is due to the way the providers search for suitability of the command, falling back to the default install for the executable when none is found. Because environment variables and commands do not refresh during the same Puppet run (due somewhat to a Windows limitation with updating environment information for currently running processes), installing to a directory that is not the default won't be detected until the next time Puppet runs. So unless you really want this installed elsewhere and don't have a current existing install in that non-default location, do not set this value.

Specifying `C:\Chocolatey` as the install directory will trigger Chocolatey to attempt to upgrade that directory. This is due to that location being the original install location for Chocolatey before it was moved to another directory and subsequently locked down. If you need this to be the installation directory, please define an environment variable `ChocolateyAllowInsecureRootDirectory` and set it to `'true'`. For more information, please see the CHANGELOG for 0.9.9.

If you override the default installation directory you need to set appropriate permissions on that install location, because Chocolatey does not restrict access to the custom directory to only Administrators. Chocolatey only restricts access to the directory in the default install location, to avoid permissions issues with custom locations, among other reasons. See "Can I install Chocolatey to another location?" for more information.

`use_7zip`
Specifies whether to use the built-in shell or allow the installer to download 7zip to extract `chocolatey.nupkg` during installation. Valid options: `true`, `false`. Default: `false`.

`choco_install_timeout_seconds`
Specifies how long in seconds should be allowed for the install of Chocolatey (including .NET Framework 4 if necessary). Valid options: Number. Default: `1500` (25 minutes).

`chocolatey_download_url`
Specifies the URL that returns `chocolatey.nupkg`. Valid options: String of URL, not necessarily from an OData feed. Any URL location will work, but it must result in the chocolatey nupkg file being downloaded. Default: `'https://chocolatey.org/api/v2/package/chocolatey/'`.

`enable_autouninstaller`
Only for 0.9.9.x users. Chocolatey 0.9.10.x+ ignores this setting. Specifies whether auto uninstaller is enabled. Auto uninstaller allows Chocolatey to automatically manage the uninstall of software from Programs and Features without necessarily requiring a `chocolateyUninstall.ps1` file in the package. Valid options: `true`, `false`. Default: `true`.

`log_output`
Specifies whether to log output from the installer. Valid options: `true`, `false`. Default: `false`.

Limitations

1. Works with Windows only.
2. If you override an existing install location of Chocolatey using `choco_install_location =>` in the Chocolatey class, it does not bring any of the existing packages with it. You will need to handle that through some other means.
3. Overriding the install location will also not allow Chocolatey to be configured or install packages on the same run that it is installed on. See `choco_install_location` for details.

Known Issues

1. This module doesn't support side by side scenarios.
2. This module may have issues upgrading Chocolatey itself using the package resource.
3. If .NET 4.0 is not installed, it may have trouble installing Chocolatey. Chocolatey version 0.9.9.9+ helps alleviate this issue.
4. If there is an error in the installer (`InstallChocolatey.ps1.erb`), it may not show as an error. This may be an issue with the PowerShell provider and is still under investigation.

Development

Puppet Inc modules on the Puppet Forge are open projects, and community contributions are essential for keeping them great. We can't access the huge number of platforms and myriad of hardware, software, and deployment configurations that Puppet is intended to serve.

We want to keep it as easy as possible to contribute changes so that our modules work in your environment. There are a few guidelines that we need contributors to follow so that we can have a chance of keeping on top of things.

For more information, see our module contribution guide.

Attributions

A special thanks goes out to Rich Siegel and Rob Reynolds who wrote the original provider and continue to contribute to the development of this provider.