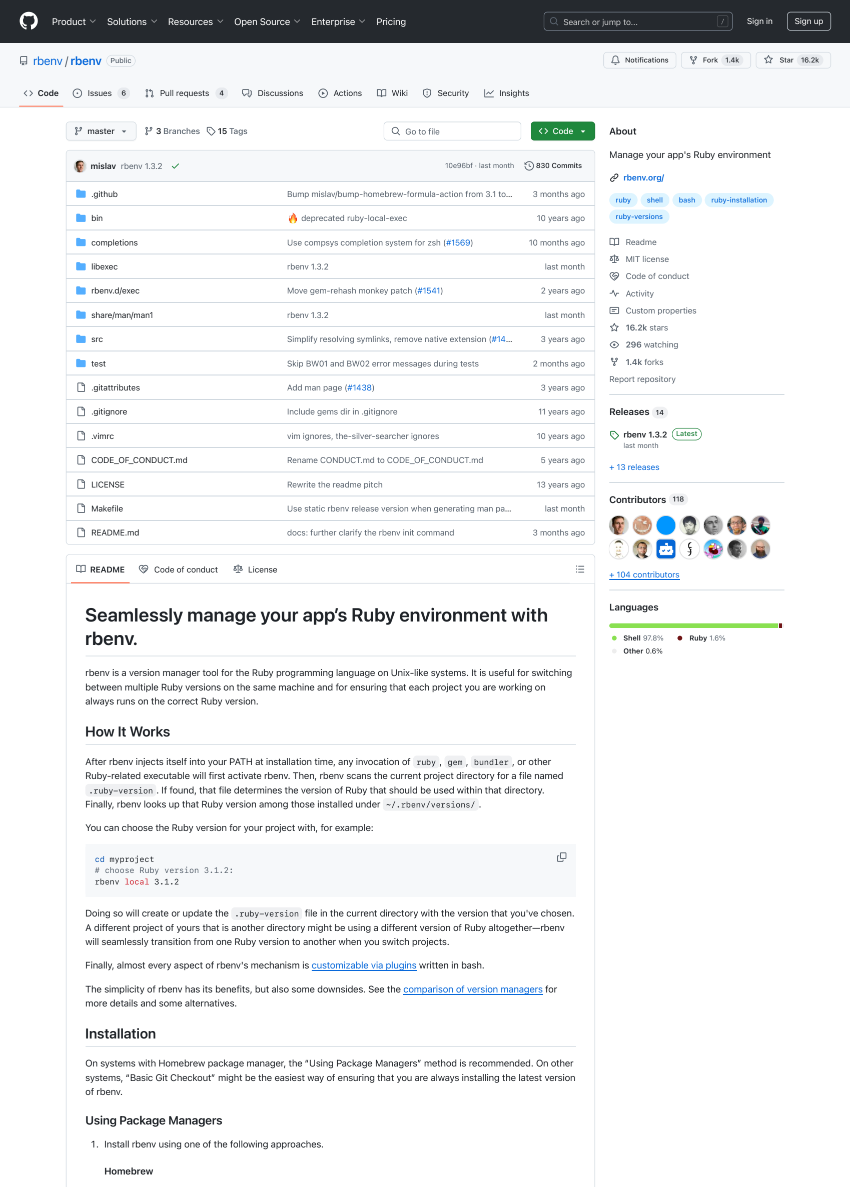

rbenv / **rbenv** Public

Notifications | Fork 1.4k | Star 16.2k

master | 3 Branches | 15 Tags | Go to file | Code

mislav rbenv 1.3.2 ✓ 10e96bf · last month 830 Commits

.github	Bump mislav/bump-homebrew-formula-action from 3.1 to...	3 months ago
bin	🔥 deprecated ruby-local-exec	10 years ago
completions	Use compsys completion system for zsh (#1569)	10 months ago
libexec	rbenv 1.3.2	last month
rbenv.d/exec	Move gem-rehash monkey patch (#1541)	2 years ago
share/man/man1	rbenv 1.3.2	last month
src	Simplify resolving symlinks, remove native extension (#14...	3 years ago
test	Skip BW01 and BW02 error messages during tests	2 months ago
.gitattributes	Add man page (#1438)	3 years ago
.gitignore	Include gems dir in .gitignore	11 years ago
.vimrc	vim ignores, the-silver-searcher ignores	10 years ago
CODE_OF_CONDUCT.md	Rename CONDUCT.md to CODE_OF_CONDUCT.md	5 years ago
LICENSE	Rewrite the readme pitch	13 years ago
Makefile	Use static rbenv release version when generating man pa...	last month
README.md	docs: further clarify the rbenv init command	3 months ago

README | Code of conduct | License

Seamlessly manage your app's Ruby environment with rbenv.

rbenv is a version manager tool for the Ruby programming language on Unix-like systems. It is useful for switching between multiple Ruby versions on the same machine and for ensuring that each project you are working on always runs on the correct Ruby version.

How It Works

After rbenv injects itself into your PATH at installation time, any invocation of `ruby`, `gem`, `bundler`, or other Ruby-related executable will first activate rbenv. Then, rbenv scans the current project directory for a file named `.ruby-version`. If found, that file determines the version of Ruby that should be used within that directory. Finally, rbenv looks up that Ruby version among those installed under `~/.rbenv/versions/`.

You can choose the Ruby version for your project with, for example:

```
cd myproject
# choose Ruby version 3.1.2:
rbenv local 3.1.2
```

Doing so will create or update the `.ruby-version` file in the current directory with the version that you've chosen. A different project of yours that is another directory might be using a different version of Ruby altogether—rbenv will seamlessly transition from one Ruby version to another when you switch projects.

Finally, almost every aspect of rbenv's mechanism is customizable via plugins written in bash.

The simplicity of rbenv has its benefits, but also some downsides. See the comparison of version managers for more details and some alternatives.

Installation

On systems with Homebrew package manager, the "Using Package Managers" method is recommended. On other systems, "Basic Git Checkout" might be the easiest way of ensuring that you are always installing the latest version of rbenv.

Using Package Managers

1. Install rbenv using one of the following approaches.

   Homebrew

About

Manage your app's Ruby environment

rbenv.org/

ruby shell bash ruby-installation ruby-versions

Readme
MIT license
Code of conduct
Activity
Custom properties
16.2k stars
296 watching
1.4k forks
Report repository

Releases 14

rbenv 1.3.2 Latest
last month

+ 13 releases

Contributors 118

+ 104 contributors

Languages

Shell 97.8% **Ruby** 1.6%
Other 0.6%

   On macOS or Linux, we recommend installing rbenv with Homebrew.

   ```
   brew install rbenv
   ```

   #### Debian, Ubuntu, and their derivatives

   > [!CAUTION]
   > The version of rbenv that is packaged and maintained in official Debian and Ubuntu repositories is *out of date*. To install the latest version, it is recommended to install rbenv using git.

   ```
   sudo apt install rbenv
   ```

   #### Arch Linux and its derivatives

   Archlinux has an AUR Package for rbenv and you can install it from the AUR using the instructions from this wiki page.

   #### Fedora

   Fedora has an official package which you can install:

   ```
   sudo dnf install rbenv
   ```

2. Set up your shell to load rbenv.

   ```
   rbenv init
   ```

3. Close your Terminal window and open a new one so your changes take effect.

That's it! You are now ready to install some Ruby versions.

Basic Git Checkout

> ⓘ Note
>
> For a more automated install, you can use rbenv-installer. If you do not want to execute scripts downloaded from a web URL or simply prefer a manual approach, follow the steps below.

This will get you going with the latest version of rbenv without needing a system-wide install.

1. Clone rbenv into `~/.rbenv`.

   ```
   git clone https://github.com/rbenv/rbenv.git ~/.rbenv
   ```

2. Set up your shell to load rbenv.

   ```
   ~/.rbenv/bin/rbenv init
   ```

   If you are curious, see here to understand what `init` does.

3. Restart your shell so that these changes take effect. (Opening a new terminal tab will usually do it.)

Shell completions

When *manually* installing rbenv, it might be useful to note how completion scripts for various shells work. Completion scripts help with typing rbenv commands by expanding partially entered rbenv command names and option flags; typically this is invoked by pressing `Tab` key in an interactive shell.

- The **bash** completion script for rbenv ships with the project and gets loaded by the `rbenv init` mechanism.

- The **zsh** completion script ships with the project, but needs to be added to FPATH in zsh before it can be discovered by the shell. One way to do this would be to edit `~/.zshrc`:

  ```
  # assuming that rbenv was installed to `~/.rbenv`
  FPATH=~/.rbenv/completions:"$FPATH"

  autoload -U compinit
  compinit
  ```

- The **fish** completion script for rbenv ships with the fish shell itself and is not maintained by the rbenv project.

Installing Ruby versions

The `rbenv install` command does not ship with rbenv out-of-the-box, but is provided by the ruby-build plugin.

Before attempting to install Ruby, **check that your build environment has the necessary tools and libraries.** Then:

```
# list latest stable versions:
rbenv install -l

# list all local versions:
rbenv install -L

# install a Ruby version:
rbenv install 3.1.2
```

For troubleshooting `BUILD FAILED` scenarios, check the ruby-build Discussions section.

> ⓘ Note
>
> If the `rbenv install` command wasn't found, you can install ruby-build as a plugin:
>
> ```
> git clone https://github.com/rbenv/ruby-build.git "$(rbenv root)"/plugins/ruby-build
> ```

Set a Ruby version to finish installation and start using Ruby:

```
rbenv global 3.1.2   # set the default Ruby version for this machine
# or:
rbenv local 3.1.2    # set the Ruby version for this directory
```

Alternatively to the `rbenv install` command, you can download and compile Ruby manually as a subdirectory of `~/.rbenv/versions`. An entry in that directory can also be a symlink to a Ruby version installed elsewhere on the filesystem.

Installing Ruby gems

Select a Ruby version for your project using `rbenv local 3.1.2`, for example. Then, proceed to install gems as you normally would:

```
gem install bundler
```

> ⓘ Note
>
> You *should not use sudo* to install gems. Typically, the Ruby versions will be installed under your home directory and thus writeable by your user. If you get the “you don't have write permissions” error when installing gems, it's likely that your "system" Ruby version is still a global default. Change that with `rbenv global <version>` and try again.

Check the location where gems are being installed with `gem env`:

```
gem env home
# => ~/.rbenv/versions/<version>/lib/ruby/gems/...
```

Uninstalling Ruby versions

As time goes on, Ruby versions you install will accumulate in your `~/.rbenv/versions` directory.

To remove old Ruby versions, simply `rm -rf` the directory of the version you want to remove. You can find the directory of a particular Ruby version with the `rbenv prefix` command, e.g. `rbenv prefix 2.7.0`.

The ruby-build plugin provides an `rbenv uninstall` command to automate the removal process.

Command Reference

The main rbenv commands you need to know are:

rbenv versions

Lists all Ruby versions known to rbenv, and shows an asterisk next to the currently active version.

```
$ rbenv versions
  1.8.7-p352
  1.9.2-p290
* 1.9.3-p327 (set by /Users/sam/.rbenv/version)
  jruby-1.7.1
  rbx-1.2.4
  ree-1.8.7-2011.03
```

rbenv version

Displays the currently active Ruby version, along with information on how it was set.

```
$ rbenv version
1.9.3-p327 (set by /Users/sam/.rbenv/version)
```

rbenv local

Sets a local application-specific Ruby version by writing the version name to a `.ruby-version` file in the current directory. This version overrides the global version, and can be overridden itself by setting the `RBENV_VERSION` environment variable or with the `rbenv shell` command.

```
rbenv local 3.1.2
```

When run without a version number, `rbenv local` reports the currently configured local version. You can also unset the local version:

```
rbenv local --unset
```

rbenv global

Sets the global version of Ruby to be used in all shells by writing the version name to the `~/.rbenv/version` file. This version can be overridden by an application-specific `.ruby-version` file, or by setting the `RBENV_VERSION` environment variable.

```
rbenv global 3.1.2
```

The special version name `system` tells rbenv to use the system Ruby (detected by searching your `$PATH`).

When run without a version number, `rbenv global` reports the currently configured global version.

rbenv shell

Sets a shell-specific Ruby version by setting the `RBENV_VERSION` environment variable in your shell. This version overrides application-specific versions and the global version.

```
rbenv shell jruby-1.7.1
```

When run without a version number, `rbenv shell` reports the current value of `RBENV_VERSION`. You can also unset the shell version:

```
rbenv shell --unset
```

Note that you'll need rbenv's shell integration enabled (step 3 of the installation instructions) in order to use this command. If you prefer not to use shell integration, you may simply set the `RBENV_VERSION` variable yourself:

```
export RBENV_VERSION=jruby-1.7.1
```

rbenv rehash

Installs shims for all Ruby executables known to rbenv (`~/.rbenv/versions/*/bin/*`). Typically you do not need to run this command, as it will run automatically after installing gems.

```
rbenv rehash
```

rbenv which

Displays the full path to the executable that rbenv will invoke when you run the given command.

```
$ rbenv which irb
/Users/sam/.rbenv/versions/1.9.3-p327/bin/irb
```

rbenv whence

Lists all Ruby versions that contain the specified executable name.

```
$ rbenv whence rackup
1.9.3-p327
jruby-1.7.1
ree-1.8.7-2011.03
```

Environment variables

You can affect how rbenv operates with the following settings:

name	default	description
`RBENV_VERSION`		Specifies the Ruby version to be used. Also see `rbenv shell`
`RBENV_ROOT`	`~/.rbenv`	Defines the directory under which Ruby versions and shims reside. Also see `rbenv root`
`RBENV_DEBUG`		Outputs debug information. Also as: `rbenv --debug <subcommand>`
`RBENV_HOOK_PATH`	*see wiki*	Colon-separated list of paths searched for rbenv hooks.
`RBENV_DIR`	`$PWD`	Directory to start searching for `.ruby-version` files.

How rbenv hooks into your shell

`rbenv init` is a helper command to hook rbenv into a shell. This helper is part of the recommended installation instructions, but optional, as an experienced user can set up the following tasks manually. The `rbenv init` command has two modes of operation:

1. `rbenv init`: made for humans, this command edits your shell initialization files on disk to add rbenv to shell startup. (Prior to rbenv 1.3.0, this mode only printed user instructions to the terminal, but did nothing else.)
2. `rbenv init -`: made for machines, this command outputs a shell script suitable to be eval'd by the user's shell.

When `rbenv init` is invoked from a bash shell, for example, it will add the following to the user's `~/.bashrc` or

`~/.bash_profile`:

```
# Added by `rbenv init` on <DATE>
eval "$(rbenv init - --no-rehash bash)"
```

You may add this line to your shell initialization files manually if you want to avoid running `rbenv init` as part of the setup process. Here is what the eval'd script does:

0. Adds `rbenv` executable to PATH if necessary.
1. Prepends `~/.rbenv/shims` directory to PATH. This is basically the only requirement for rbenv to function properly.
2. Installs bash shell completion for rbenv commands.
3. Regenerates rbenv shims. If this step slows down your shell startup, you can invoke `rbenv init -` with the `--no-rehash` flag.
4. Installs the "sh" dispatcher. This bit is also optional, but allows rbenv and plugins to change variables in your current shell, making commands like `rbenv shell` possible.

Uninstalling rbenv

The simplicity of rbenv makes it easy to temporarily disable it, or uninstall from the system.

1. To **disable** rbenv managing your Ruby versions, simply comment or remove the `rbenv init` line from your shell startup configuration. This will remove rbenv shims directory from PATH, and future invocations like `ruby` will execute the system Ruby version, bypassing rbenv completely.

   While disabled, `rbenv` will still be accessible on the command line, but your Ruby apps won't be affected by version switching.

2. To completely **uninstall** rbenv, perform step (1) and then remove the rbenv root directory. This will **delete all Ruby versions** that were installed under `` `rbenv root`/versions/ ``:

   ```
   rm -rf "$(rbenv root)"
   ```

   If you've installed rbenv using a package manager, as a final step perform the rbenv package removal:

   - Homebrew: `brew uninstall rbenv`
   - Debian, Ubuntu, and their derivatives: `sudo apt purge rbenv`
   - Archlinux and its derivatives: `sudo pacman -R rbenv`

Development

Tests are executed using Bats:

```
$ bats test
$ bats test/<file>.bats
```

Please feel free to submit pull requests and file bugs on the issue tracker.