RubyGems Ruby package manager

START HERE

`gem install <gem>`
Install a gem

`gem update`
Update installed gems

`gem list`
Installed gems

`gem uninstall <gem>`
Remove a gem

`gem search <regexp>`
Find gems on the server

`gem build <name>.gemspec`
Package your own gem

`gem info <gem>`
Details of one gem

`gem env`
Where gems live

Gemspec your gem's metadata → `gem build` → **.gem file** packaged archive → `gem push` → **rubygems.org** the default source → `gem install` → **GEM_HOME** installed gems → `require` → **Your Ruby code**

CONCEPTS

gem installs gems machine-wide. For a project, list them in a Gemfile and use bundle (see Bundler) so everyone gets the same versions.

Without write access to the system Ruby, gem install fails with a permissions error. Use --user-install, or a Ruby version manager that keeps gems in your home.

Rubies from a version manager keep their own gem set, so a gem installed under one Ruby is missing under another.

INSTALL

`gem install <gem>`
Install the latest version

`gem install <gem> -v "~> 1.2"`
Install a version matching a constraint

`gem install <gem> --user-install`
Install into your home directory
Binaries then land in the user gem bin dir; add it to PATH (gem env shows the path).

`gem install <gem> --no-document`
Skip generating docs (much faster)

`gem install <gem> --prerelease`
Allow prerelease versions

`gem install <gem> -s <url>`
Fetch from another source

`gem install -g Gemfile`
Install the gems a Gemfile lists

`gem install <name>.gem`
Install a local .gem file

`gem install <gem> --conservative`
Keep already-installed dependencies

`gem fetch <gem>`
Download the .gem file without installing

`gem exec <cmd>`
Run a gem's executable, installing if needed

LIST & SEARCH

`gem list`
List installed gems

LIST & SEARCH (cont.)

`gem list <regexp>`
Filter installed gems by name

`gem list -d`
Include description and details

`gem list <gem> -i`
Is it installed? Prints true or false

`gem search <regexp>`
Search remote gems by name

`gem search <regexp> -ae`
Exact name, all versions

`gem list -r <regexp>`
List remote gems

`gem info <gem>`
Version, license, homepage, install path

`gem contents <gem>`
List an installed gem's files

`gem which <file>`
Path of a requireable file

`gem dependency <gem>`
Show a gem's dependencies

`gem dependency <gem> -R`
Show what depends on it

`gem specification <gem>`
Show the full gemspec

`gem specification <gem> <field>`
Show a single gemspec field

UPDATE & REMOVE

`gem outdated`
List gems with newer versions

`gem update`
Update all installed gems
Older versions stay installed; run gem cleanup to remove them.

`gem update <gem>`
Update one gem

`gem update --system`
Update RubyGems itself

`gem update --system <version>`
Move RubyGems to a given version

`gem uninstall <gem>`
Remove a gem

`gem uninstall <gem> -v <version>`
Remove one version

`gem uninstall <gem> -a`
Remove all versions without asking

`gem uninstall <gem> -x`
Also remove its executables

UPDATE & REMOVE (cont.)

`gem uninstall <gem> -I`
Ignore dependency checks

`gem cleanup`
Remove old versions of gems

`gem cleanup -n`
Dry run: show what would go

`gem pristine <gem>`
Restore a gem to its packaged state

`gem pristine --all`
Restore every installed gem

SOURCES

`gem sources`
Show configured sources

`gem sources -a <url>`
Add a source

`gem sources -r <url>`
Remove a source

`gem sources -u`
Refresh the source cache

`gem sources --prepend <url>`
Add a source at the top

BUILD & PUBLISH

`gem build <name>.gemspec`
Package a gem into a .gem file

`gem build <name>.gemspec -o <file>`
Choose the output file name

`gem build <name>.gemspec --strict`
Fail on gemspec warnings

`gem signin`
Sign in and store an API key

`gem push <name>-<version>.gem`
Upload a gem to rubygems.org

`gem push <file> --otp <code>`
Push with a one-time MFA code

`gem push <file> --host <url>`
Push to another gem server

`gem owner <gem>`
List the owners of a gem

`gem owner <gem> -a <email>`
Add an owner

`gem yank <gem> -v <version>`
Pull a pushed version from the index
A yank cannot be undone and the version number cannot be reused.

BUILD & PUBLISH (cont.)

`gem signout`
Sign out of all sessions

`gem unpack <gem>`
Extract a gem into the current folder

`gem open <gem>`
Open a gem's source in your editor

ENVIRONMENT

`gem env`
Show RubyGems configuration

`gem env home`
Where gems are installed

`gem env gempath`
All directories searched for gems

`gem env version`
RubyGems version

`gem --version`
Print the RubyGems version

`gem help <command>`
Help for one command

`gem help commands`
List every command

`gem check --doctor`
Find and clean orphaned gem files

`GEM_HOME=<dir> gem install <gem>`
Install into a custom directory

Version constraints (-v)

Constraint	Meaning
`"1.2.3"`	exactly 1.2.3
`"⩾ 1.2"`	1.2 or newer
`"~> 1.2"`	1.2 up to below 2.0
`"~> 1.2.3"`	1.2.3 up to below 1.3
`"< 2"`	anything below 2

RECIPES

Install without docs, per user

```
gem install <gem> --user-install
--no-document
# put the user bin dir on PATH
gem env gempath
```

Tidy a gem set

```
gem outdated
gem update
gem cleanup -n
gem cleanup
```

Publish a gem

```
gem build <name>.gemspec
gem signin
gem push <name>-<version>.gem
```

Debug a missing gem

```
gem list <gem>
gem which <file>
gem env gempath
gem pristine <gem>
```