



Guides

Bundler in gems
Frequently Asked Questions
Gemfiles
Getting Started
How to Upgrade to Bundler 2
How to create a Ruby gem with Bundler
How to deploy bundled applications
How to install gems from git repositories
How to manage application dependencies with Bundler
How to manage groups of gems
How to package and share code using a Gemfile
How to troubleshoot RubyGems and Bundler TLS/SSL Issues
How to update gems with Bundler
How to use Bundler in a single-file Ruby script
How to use Bundler with Docker
How to use Bundler with Rails
How to use Bundler with Ruby
How to use Bundler with RubyMotion
How to use Bundler with Sinatra
How to use git bisect with Bundler
How to write a Bundler plugin
Known Plugins
Recommended Workflow with Version Control
Ruby Directive
Why Bundler exists

Using Bundler while developing a gem

If you’re creating a gem from scratch, you can use bundler’s built in gem skeleton to create a base gem for you to edit.

```
$ bundle gem my_gem
```

This will create a new directory named `my_gem` with your new gem skeleton. If you already have a gem, you can create a Gemfile and use Bundler to manage your development dependencies. Here’s an example.

```
source "https://rubygems.org"
gemspec
gem "rspec", "~> 3.9"
gem "rubocop", "0.79.0"
```

In this Gemfile, the `gemspec` method imports gems listed with `add_runtime_dependency` in the `my_gem.gemspec` file, and it also installs rspec and rubocop to test and develop the gem. All dependencies from the gemspec and Gemfile will be installed by `bundle install`, but rspec and rubocop will not be included by `gem install mygem` or `bundle add mygem`. Runtime dependencies in your gemspec are treated as if they are listed in your Gemfile, and development dependencies are added by default to the group, `:development`. You can change that group with the `:development_group` option

```
gemspec :development_group => :dev
```

As well, you can point to a specific gemspec using `:path`. If your gemspec is in `/gemspec/path`, use

```
gemspec :path => '/gemspec/path'
```

If you have multiple gemspecs in the same directory, specify which one you’d like to reference using `:name`

```
gemspec :name => 'my_awesome_gem'
```

This will use `my_awesome_gem.gemspec` That’s it! Use bundler when developing your gem, and otherwise, use gemspecs normally!

```
$ gem build my_gem.gemspec
```

Edit this document on GitHub if you caught an error or noticed something was missing.