**In Depth**

Read the manual for an in-depth discussion of all of the options available in the Gemfile and how to use them.

[Gemfile manual](https://bundler.io/man/gemfile.5.html)

**Gemfiles**

Gemfiles require at least one gem source, in the form of the URL for a RubyGems server. Generate a Gemfile with the default rubygems.org source by running bundle init. If you can, use https so your connection to the rubygems.org server will be verified with SSL.

source 'https://rubygems.org'

*# Gems here*

If you want to pick up gems from a different source, make sure to wrap the extra sources in a block. Multiple global source lines are a security risk and should not be used as they can lead to gems being installed from unintended sources.

source 'https://rubygems.org'

*# Gems here*

source 'https://gems.example.com' do

*# Gems from the alternative source here*

end

Some gem sources require a username and password. Use bundle config to set the username and password for any sources that need it. The command must be run once on each computer that will install the Gemfile, but this keeps the credentials from being stored in plain text in version control.

$ bundle config https://gems.example.com/ user:password

For some sources, like a company Gemfury account, it may be easier to simply include the credentials in the Gemfile as part of the source URL.

source "https://user:password@gems.example.com"

Credentials in the source URL will take precedence over credentials set using config.

Declare the gems that you need, including version numbers. Specify versions using the same syntax that RubyGems supports for dependencies.

gem 'nokogiri'

gem 'rails', '5.0.0'

gem 'rack', '>=1.0'

gem 'thin', '~>1.1'

Most of the version specifiers, like >= 1.0, are self-explanatory. The specifier ~> has a special meaning, best shown by example. ~> 2.0.3 is identical to >= 2.0.3 and < 2.1. ~> 2.1 is identical to >= 2.1 and < 3.0. ~> 2.2.beta will match prerelease versions like 2.2.beta.12. ~> 0 is identical to >= 0.0 and < 1.0.

[RubyGems version specifiers](http://guides.rubygems.org/patterns/#pessimistic-version-constraint)

If a gem's main file is different than the gem name, specify how to require it.

gem 'rack-cache', :require => 'rack/cache'

gem 'sqlite3'

Specify :require => false to prevent bundler from requiring the gem, but still install it and maintain dependencies.

gem 'rspec', :require => false

gem 'sqlite3'

In order to require gems in your Gemfile, you will need to call Bundler.require in your application.

[Learn More: Bundler.require](https://bundler.io/guides/groups.html)

If some of your gems need to be fetched from a private gem server, this default source can be overridden for those gems. For a gem server that contains a single gem, it is easiest to use the :source option on that gem.

gem 'my\_gem', '1.0', :source => 'https://gems.example.com'

If several gems come from the same server, you can use a source block to group them together.

source 'https://gems.example.com' do

gem 'my\_gem', '1.0'

gem 'another\_gem', '1.2.1'

end

Credentials for gem servers can be specified either in the URL or using bundle config, as described above.

Git repositories are also valid gem sources, as long as the repo contains one or more valid gems. Specify what to check out with :tag, :branch, or :ref. The default is the master branch.

gem 'nokogiri', :git => 'https://github.com/tenderlove/nokogiri.git', :branch => '1.4'

If the git repository does not contain a .gemspec file, bundler will create a simple one, without any dependencies, executables or C extensions. This may work for simple gems, but not work for others. If there is no .gemspec, you probably shouldn't use the gem from git.

[Learn More: Git](https://bundler.io/guides/git.html)

If you would like to use an unpacked gem directly from the filesystem, simply set the :path option to the path containing the gem's files.

gem 'extracted\_library', :path => './vendor/extracted\_library'

If you would like to use multiple local gems directly from the filesystem, you can set a global `path` option to the path containing the gem's files. This will automatically load gemspec files from subdirectories.

path 'components' do

gem 'admin\_ui'

gem 'public\_ui'

end

Dependencies can be placed into groups. Groups can be ignored at install-time (using --without) or required all at once (using Bundler.require).

gem 'wirble', :group => :development

gem 'debugger', :group => [:development, :test]

group :test do

gem 'rspec'

end

[Learn More: Groups](https://bundler.io/guides/groups.html)

You can specify the required version of Ruby in the Gemfile with ruby. If the Gemfile is loaded on a different Ruby version, Bundler will raise an exception with an explanation.

ruby '1.9.3'

What this means is that this app has a dependency to a Ruby VM that is ABI compatible with 1.9.3. If the version check does not match, Bundler will raise an exception. This will ensure the running code matches. You can be more specific with the :engine and :engine\_version options.

ruby '1.9.3', :engine => 'jruby', :engine\_version => '1.6.7'