

Ruby - Feature #20309

Bundled gems for Ruby 3.5

02/27/2024 10:16 PM - hsbt (Hiroshi SHIBATA)

Status:Closed Priority:Normal Assignee:hsbt (Hiroshi SHIBATA) Target version:	
Description I propose migrate the following default gems to bundled gems at Ruby 3.5. So, It means users will get warnings if users try to load them. (Update with 2024/03/14, 2024/06/05, 2024/09/06) • rdoc(done) ◦ We need to change build task like download rdoc gem before document generation. ▪ extract make doc from make all and invoke make doc before make install. • done for Ruby 3.4 ◦ or We make document generation is optional from Ruby 3.5 ▪ We explicitly separate make install and make install-doc • ostruct(done) ◦ I make ostruct as optional on json at https://github.com/flori/json/pull/565 • pstore(done) • win32ole(done) • logger(done) ◦ activesupport needs to add logger to its dependency same as bigdecimal, drb or etc. • fiddle(done) • benchmark • irb ◦ We need to consider how works binding.irb after Ruby 3.5. ◦ I consider to use irb without Gemfile. • reline • readline (wrapper file for readline-ext and reline) I have a plan to migrate the following default gems too. But I need to more feedback from other committers about them. • io-console ◦ rubygems uses that. Should we make optional that? • open-uri • yaml (wrapper file for psych) ◦ syck is retired today. I'm not sure what people uses psych directly, not yaml. • un ◦ ruby -run is one of cool feature of Ruby. Should we avoid uninstalling un gem? ◦ mkmf uses ruby -run for that. I need to investigate that. • singleton ◦ This is famous design pattern. Should we enforce users add them to their Gemfile? • forwardable ◦ reline needs to add forwardable their runtime_dependency after migration. • weakref ◦ I'm not sure how impact after migrating bundled gems. • fcntl ◦ Should we integrate these constants into ruby core? I would like to migrate ipaddr and uri too. But these are used by webrick that is mock server for our test suite. We need to rewrite webrick with TCPSocker or extract ipaddr and uri dependency from webrick Other default gems depend on our build process or other libraries deeply. I will update this proposal if I could extract them from default gems.	
Related issues: Related to Ruby - Feature #20187: Bundled gems at Ruby 3.4 Closed Related to Ruby - Feature #19351: Promote bundled gems at Ruby 3.3 Closed	

Related to Ruby - Feature #20347: Separate docs task from all	Closed
Related to Ruby - Bug #20714: Handle optional dependencies in `bundled_gems.rb`	Assigned
Related to Ruby - Feature #21442: Make tsort to bundled gems	Assigned

History

#1 - 02/27/2024 10:39 PM - rubyFeedback (robert heiler)

ruby -run is one of cool feature of Ruby. Should we avoid uninstalling un gem?

I think -run is kind of neat; it's a bit like a mini-DSL for the commandline.

(Having said that, I actually do not use it myself; I instead use a custom executable that calls methods in other .rb files, so I kind of have the same or similar functionality available without needing to use special commandline flags; and lots of aliases.)

#2 - 02/28/2024 12:48 AM - hsbt (Hiroshi SHIBATA)

- Description updated

#3 - 02/28/2024 12:49 AM - hsbt (Hiroshi SHIBATA)

- Description updated

#4 - 02/28/2024 04:28 AM - hsbt (Hiroshi SHIBATA)

- Description updated

#5 - 02/28/2024 08:46 AM - retro (Josef Šimánek)

I'm big fan (and user) of ruby -run -e httpd . -p 1234, this is also part of "local HTTP server oneliners" like <https://gist.github.com/willurd/5720255> and it will break various tutorials online if un is not installed with ruby by default.

#6 - 02/28/2024 11:14 AM - Eregon (Benoit Daloze)

hsbt (Hiroshi SHIBATA) wrote:

- ostruct
 - I make ostruct as optional on json at <https://github.com/flori/json/pull/565>

+1 since ostruct is already kind of deprecated.

- irb
 - We need to consider how works binding.irb after Ruby 3.5.

Will that break binding.irb under bundle exec? If so that seems a major problem.

I would think it would break it if binding.irb is implemented as e.g. gem "irb"; require "irb"; binding.irb.

- yaml (wrapper file for psych)
 - syck is retired today. I'm not sure what people uses psych directly, not yaml.

psych seems to be a default gem (and I guess will remain as so as a dependency of rubygems).

If so, what is the advantage to make yaml a bundled gem?

It's a [trivial wrapper](#) so I don't see advantage to make bundled instead of default gem, only overhead (adding gem "yaml" in Gemfiles which has no value as this code almost never changes) and troubles.

BTW I think most programs use require "yaml" instead of require "psych" and I think that's better as it is, psych is kind of an implementation detail, people want to load/dump YAML.

It seems busy unproductive work to replace require "yaml" by require "psych" or to add gem "yaml" in Gemfiles, so I am very much against making yaml a bundled gem.

- readline (wrapper file for readline-ext and reline)

Is the intention to have people do require "reline" instead of require "readline"?

It seems weird if people add gem "readline" to solve this, when gem "reline" would make much more sense.

It's a [trivial wrapper](#), so I think making it a bundled gem creates more problems/overhead than it solves.

- un

- ruby -run is one of cool feature of Ruby. Should we avoid uninstalling un gem?

ruby -run would still work if un is a bundled gem, so I think that's OK.

Maybe RubyGems should ask confirmation/extra warning when uninstalling a bundled gem.

- singleton
 - This is famous design pattern. Should we enforce users add them to their Gemfile?

IMO not worth it, it's pretty [trivial code](#) and too much overhead to ask users to add to Gemfile for something so basic.

- weakref
 - I'm not sure how impact after migrating bundled gems.

[weakref](#) is very tightly bound to implementation details, such as `::ObjectSpace::WeakMap.new` on CRuby.

The implementation is completely different on JRuby and TruffleRuby.

It will cause problems to make this a bundled gem for alternative Ruby implementations.

So I think it is not worth to make this a bundled gem.

Also, that file is tiny and trivial, and had [very few changes](#).

- fcntl
 - Should we integrate these constants into ruby core?

It seems low-level stuff so I think being behind a require is in general good.

But between core and bundled gem I would prefer core.

BTW the reasoning to change from default gem to bundled gem is at <https://bugs.ruby-lang.org/issues/19351#note-16>

#7 - 02/28/2024 11:18 AM - Eregon (Benoit Daloze)

Eregon (Benoit Daloze) wrote in [#note-6](#):

- fcntl
 - Should we integrate these constants into ruby core?

It seems low-level stuff so I think being behind a require is in general good.

But between core and bundled gem I would prefer core.

I misremembered, I thought there was something like `Fcntl.fcntl`, but it's `IO#fcntl`.

So `fcntl` is literally just a bunch of constants:

<https://github.com/oracle/truffleruby/blob/master/lib/truffle/fcntl.rb>

<https://github.com/ruby/ruby/blob/master/ext/fcntl/fcntl.c>

So I think this should be core then.

#8 - 02/28/2024 11:20 AM - Eregon (Benoit Daloze)

- Related to Feature #20187: Bundled gems at Ruby 3.4 added

#9 - 02/28/2024 11:21 AM - Eregon (Benoit Daloze)

- Related to Feature #19351: Promote bundled gems at Ruby 3.3 added

#10 - 02/28/2024 03:38 PM - jeremyevans0 (Jeremy Evans)

Eregon (Benoit Daloze) wrote in [#note-6](#):

hsbt (Hiroshi SHIBATA) wrote:

- singleton
 - This is famous design pattern. Should we enforce users add them to their Gemfile?

IMO not worth it, it's pretty [trivial code](#) and too much overhead to ask users to add to Gemfile for something so basic.

On the other hand, it's almost always better to use a normal constant with a singleton object than the singleton library. singleton's only advantage is you can delay initialization, and if you want that, you can use autoload for the file that defines the constant.

Instead of:

```
class Klass
  include Singleton

  # define methods
end

Klass.instance
```

Use:

```
CONSTANT = Object.new
CONSTANT.singleton_class.class_eval do
  # define methods
end
```

I think we should encourage users to use the singleton object support built into Ruby core, as opposed to a separate library that accomplishes mostly the same thing. I think the main advantage of the singleton library over standard Ruby is that it supports marshalling, but I'm not sure how common is the need to marshal singleton objects.

I'm in favor of moving the singleton library to bundled gems mainly to discourage users from using it.

#11 - 02/28/2024 05:21 PM - Eregon (Benoit Daloze)

[@jeremyevans0 \(Jeremy Evans\)](#) One issue with that replacement is it does not name the class of the object, so it's quite unclear if there is an exception with it.

And also no easy way to define constants in that singleton class (and if one does `A = ...` there it will accidentally declare `Object::A`).

I think there is nothing wrong with the singleton stdlib, so I think we should not discourage using it.

There are cases where it's best to initialize a constant eagerly and that's fine, those usages do not need singleton.

#12 - 02/28/2024 05:23 PM - Eregon (Benoit Daloze)

Also something I remembered now is YJIT [does not compile singleton methods](#), except singleton methods of modules and classes. So then that replacement can also be significantly slower.

#13 - 03/14/2024 04:54 AM - hsbt (Hiroshi SHIBATA)

- Description updated

#14 - 03/14/2024 09:44 AM - hsbt (Hiroshi SHIBATA)

Thanks [@Eregon \(Benoit Daloze\)](#) and [@jeremyevans0 \(Jeremy Evans\)](#).

I mostly agreed your comments. And I discussed this at DevMeeting 2024/03/14.

- No one against about ostruc. I will do that.
- We should consider to run `irb` without gem "`irb`" of Gemfile under the all of Bundler environment.
 - I will consider it with `irb`, `reline` and `io-console`.
- I try to run `make doc` before `make install` and use `rdoc` as bundled gems.
 - If I can do that, I will mark `rdoc` as bundled gems at Ruby 3.5.
- I'll extract another issue for singleton, `un`, `fcntl`.

I will update this issue until finalised target libraries.

#15 - 03/14/2024 10:41 AM - Eregon (Benoit Daloze)

[@hsbt \(Hiroshi SHIBATA\)](#) What about [yaml](#) and [readline](#)?

I think these are not worth moving to bundled gems, i.e. the gains are (AFAIK) very small while the costs and confusion are high.

I think so because the code for these 2 trivial "loaders/wrappers" (see links above) almost never changes (so the overhead of keeping them in sync is basically 0, probably 0 security issues in that code, etc).

There seems to be no value to use track `readline` 0.0.4 vs `readline` 0.0.5 in a `Gemfile.lock` for instance, as the only likely changes would be to adapt to some incompatible change in Ruby, and then that's useless since the right version would be shipped with Ruby.

The main problem there/my main concern is the cost, many people would need to add gem "`readline`" or gem "`yaml`" to their Gemfile for Ruby 3.5, and gain basically nothing from it, so it would be almost pure overhead.

Also depending on gem "`yaml`" alone is not a good idea, because it doesn't mean any specific major version of `psych`. So one needs gem "`psych`" anyway, and the gem "`yaml`" is redundant.

To try to quantify that I looked at

- https://rubygems.org/gems/readline/reverse_dependencies: only 5 gems depend on `readline` right now. Yet there are 371+1158 matches for `gem-codesearch 'require "readline"' + gem-codesearch "require 'readline'".` All these gems would need to add gem "`readline`", for no gain.
- https://rubygems.org/gems/psych/reverse_dependencies vs https://rubygems.org/gems/yaml/reverse_dependencies. There are about 18x

(54M/3M) as much downloads for psych than yaml. RubyGems.org doesn't show a total count for reverse dependencies but it seems clear few gems depend on yaml, much much more depend on psych. All the gems depending on psych and not yaml would need changes.

One more thing, as far as I can see, psych is a default gem. Having yaml as a bundled gem while psych is a default gem would lead to an awkward situation that when run under bundler and not adding either to the Gemfile, one can require "psych" but cannot require "yaml". So they could use Psych in code but not YAML which feels a wrong limitation (people might just do YAML=Psych themselves as a workaround, but then that could cause warnings).

So it seems clear yaml shouldn't become bundled gem before psych at least. And I think psych cannot become a bundled gem because RubyGems needs it.

#16 - 03/14/2024 10:55 AM - hsbt (Hiroshi SHIBATA)

What about yaml and readline?

There is no conclusion yet. I understood your concern. Do not rush this.

#17 - 03/14/2024 02:58 PM - Eregon (Benoit Daloze)

OK, thank you. I wanted to make sure my concern on that is clear.

At least it seems useful to have some data and extra details on it (the last paragraph).

#18 - 03/14/2024 10:36 PM - hsbt (Hiroshi SHIBATA)

At least it seems useful to have some data and extra details on it (the last paragraph).

I see. Thanks for your investigation.

To be precise, there wasn't time to talk about wrapper files. We only discuss about <https://bugs.ruby-lang.org/issues/20309#note-14> yesterday.

#19 - 03/19/2024 09:15 AM - hsbt (Hiroshi SHIBATA)

- *Related to Feature #20347: Separate docs task from all added*

#20 - 06/04/2024 06:24 AM - hsbt (Hiroshi SHIBATA)

- *Description updated*

#21 - 06/05/2024 08:27 AM - hsbt (Hiroshi SHIBATA)

- *Description updated*

#22 - 06/09/2024 11:47 AM - Eregon (Benoit Daloze)

I wonder if making win32ole a bundled gem is a good idea.

From what I have seen the consequence of such changes seems in most cases to encourage gems to drop the dependency on the newly-bundled gem.

That may be good for e.g. ostruct which is kind of self-deprecated (due to horrible performance and messy semantics).

But it may be bad for win32ole, because the alternatives may be slower or less reliable, e.g. see the discussion [here](#) where it's all too easy to regress from 60ms to 1s, and potentially fail if neither powershell nor wmic are available.

I don't think many gems will consider adding win32ole as a dependency, because it looks very weird for usages of the gem on non-Windows.

Even though the gem does seem to install OK on non-Windows, when installing CRuby on non-Windows, win32ole has never been installed as a default gem, so it looks unexpected to even install that gem.

#23 - 08/29/2024 09:37 PM - hsbt (Hiroshi SHIBATA)

- *Description updated*

#24 - 08/29/2024 10:39 PM - Eregon (Benoit Daloze)

Regarding whether to make benchmark a bundled gem I think there are some downsides (from <https://github.com/ruby/ruby/pull/11492>):

- I think it's valuable that Benchmark.realtime { ... } is available without needing a gem dependency. It's a convenient thing, in gems, in irb, when performance debugging, etc. Having to add a gem to the Gemfile/gemspec just for that is inconvenient. It starts to feel a bit like JS/npm when one needs a dependency for something so simple (or copy/paste the code around, which can be suboptimal e.g. Benchmark.realtime could be implemented internally more efficiently or e.g. it changed from Time.now to Process.clock_gettime). Also Benchmark.measure is less simple yet still convenient.
- Every gem using Benchmark will have to choose whether to add a dependency on the benchmark gem (which seems a bit heavy given how

small it is) or duplicating the code of benchmark.rb. The PRs linked at <https://github.com/ruby/ruby/pull/11492> are a good illustration of the effect of making such small gems bundled: code duplication across many gems, if there is ever an issue with that copied code (unlikely in this case I admit) it will be that much harder to fix it.

- I don't think there has ever been a security issue in benchmark.rb or a need for a CRuby release for benchmark.rb. It doesn't change much either, so sync issues don't seem a big deal here. So the gains to make it bundled instead of default seems very small (or am I missing something?).

#25 - 08/30/2024 04:06 PM - Earlopain (Earlopain _)

I guess I can share my opinion here.

There definitely is a benefit in bundling some gems, both from a ruby maintainer and security perspective. But with things like benchmark, forwardable, singleton, (or base64, mutex_m from the previous issue) I am hardpressed to actually include these into the dependency graph.

Many libraries depend on them, and usually the usage is trivial. For base64, it's just a different, though less beautifully named method invocation (if you don't use the url-safe variants), singleton and forwardable are what I would consider syntactic sugar. I'm not going to add a dependency just for that, I will instead forgoe the small ergonomics I'd gain and go with the manual solution.

Note this is purely from a library perspective. End consumers are free to do what they want but if I can do the same thing myself in maybe 1 or 2 lines more I'm just going to do that. From PRs I've opened against other projects, maintainers tend to agree. Adding a dependency just for that usually isn't justifiable.

#26 - 09/05/2024 11:24 AM - hsbt (Hiroshi SHIBATA)

- Related to Bug #20714: Handle optional dependencies in `bundled\_gems.rb` added

#27 - 09/06/2024 01:58 AM - hsbt (Hiroshi SHIBATA)

- Description updated

#28 - 09/06/2024 01:59 AM - hsbt (Hiroshi SHIBATA)

- Description updated

#29 - 09/06/2024 05:35 AM - hsbt (Hiroshi SHIBATA)

I will warn benchmark, irb and reline at Ruby 3.4. and make them and readline wrapper to bundled gems at Ruby 3.5 with September Dev Meeting. The current list is final proposal for Ruby 3.4.

I know bundled_gems.rb have some incomplete issues:

- Ignore to warn for declared dependencies like reline at irb.
- Suppress warning for optional dependency
 - Related with <https://github.com/ruby/ruby/pull/11550> or <https://github.com/ruby/ruby/pull/11545>

To Earlopain.

singleton and forwardable are what I would consider syntactic sugar. I'm not going to add a dependency just for that

Agreed. I withdraw them like singleton to make bundled gems.

#30 - 09/06/2024 02:48 PM - ima1zumi (Mari Imaizumi)

Does this change mean that even if Ruby is installed, the irb command will no longer be available?

If that is the case, I oppose it for the following reasons:

- Existing learning materials will no longer work as-is, causing confusion for beginners.
- IRB would need to be reinstalled every time Ruby is reinstalled.

I believe removing IRB would significantly disadvantage users.

#31 - 09/09/2024 12:16 AM - hsbt (Hiroshi SHIBATA)

Does this change mean that even if Ruby is installed, the irb command will no longer be available?

No. irb is available as bundled gems.

#32 - 09/09/2024 02:35 AM - ima1zumi (Mari Imaizumi)

No. irb is available as bundled gems.

Sorry, I misunderstood.

#33 - 09/13/2024 05:44 PM - deivid (David Rodríguez)

Regarding irb, I find it concerning that we need to add code like this:

<https://github.com/ruby/ruby/blob/9afc6a981deae6e23d938cf5c2c4baadfeaafdb1/prelude.rb#L12-L44>. Those are internals of Bundler that are likely to break, and I can't even tell if that is doing what's intended?

The fact that we want binding.irb to work regardless of being in a bundle exec context or not feels like a sign that irb should probably stay as a default gem?

#34 - 09/16/2024 10:27 AM - Eregon (Benoit Daloze)

There is an additional issue with making fiddle a bundled gem: it means any usage now must have libffi headers available or it will fail to install. Before, it would just work because fiddle would have been built when CRuby was built.

Seen in <https://github.com/ffi/ffi/actions/runs/10881406447/job/30190385022?pr=1119>:

```
extconf.rb:78:in `<main>': missing libffi. Please install libffi or use
--with-libffi-source-dir with libffi source location. (RuntimeError)
```

From <https://github.com/ffi/ffi/pull/1119>

#35 - 09/16/2024 08:35 PM - vo.x (Vit Ondruch)

Eregon (Benoit Daloze) wrote in [#note-34](#):

There is an additional issue with making fiddle a bundled gem: it means any usage now must have libffi headers available or it will fail to install. Before, it would just work because fiddle would have been built when CRuby was built. Seen in <https://github.com/ffi/ffi/actions/runs/10881406447/job/30190385022?pr=1119>:

```
extconf.rb:78:in `<main>': missing libffi. Please install libffi or use
--with-libffi-source-dir with libffi source location. (RuntimeError)
```

From <https://github.com/ffi/ffi/pull/1119>

I don't think it is bad idea to start using bundle install --local, i.e. use gems which are already available on the system. And on top of that, stop using Bundler in containers for runtime in production use.

#36 - 09/17/2024 08:53 AM - hsbt (Hiroshi SHIBATA)

The fact that we want binding.irb to work regardless of being in a bundle exec context or not feels like a sign that irb should probably stay as a default gem?

I don't think so. Bundler should provide that feature for like debug, stackprof and irb. I and [@mame \(Yusuke Endoh\)](#) discussed about that. We sometimes need to use that tools without updating Gemfile and its lockfile.

#37 - 09/17/2024 09:40 AM - deivid (David Rodríguez)

I don't think so. Bundler should provide that feature for like debug, stackprof and irb. I and [@mame \(Yusuke Endoh\)](#) (Yusuke Endoh) discussed about that. We sometimes need to use that tools without updating Gemfile and its lockfile.

Yeah, that sounds reasonable to me, and you can already do that with default gems. My question is more, why does irb need to be converted into a bundled gem? I think the conversion does have a maintenance cost, so I want to make sure the benefits are worth that cost.

If we still want to do this, can we add some test somewhere that binding.irb works in a bundle exec context, to make sure we don't break that code in prelude.rb?

#38 - 09/17/2024 12:37 PM - Eregon (Benoit Daloze)

vo.x (Vit Ondruch) wrote in [#note-35](#):

I don't think it is bad idea to start using bundle install --local, i.e. use gems which are already available on the system. And on top of that, stop using Bundler in containers for runtime in production use.

Do you mean --prefer-local? I don't think --local would work in general since that does not attempt to connect to rubygems.org at all.
EDIT: From the docs --prefer-local seems the right thing, we should try it:

```
--local
  Do not attempt to connect to rubygems.org. Instead, Bundler will use the gems already present
in Rubygems' cache or in vendor/cache. Note that if an
appropriate platform-specific gem exists on rubygems.org it will not be found.

--prefer-local
  Force using locally installed gems, or gems already present in Rubygems' cache or in vendor/cach
e, when resolving, even if newer versions are available
  remotely. Only attempt to connect to rubygems.org for gems that are not present locally.
```

#39 - 09/18/2024 06:30 AM - vo.x (Vit Ondruch)

Eregon (Benoit Daloze) wrote in [#note-38](#):

vo.x (Vit Ondruch) wrote in [#note-35](#):

I don't think it is bad idea to start using bundle install --local, i.e. use gems which are already available on the system. And on top of that, stop using Bundler in containers for runtime in production use.

Do you mean --prefer-local? I don't think --local would work in general since that does not attempt to connect to rubygems.org at all.

I was not aware of --prefer-local (I am trying to avoid Bundler as much as I can), but yes, that sounds as reasonable alternative.

#40 - 02/06/2025 09:22 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Assigned to Closed

I migrated all of target gems to bundled gems in master branch.

[@k0kubun \(Takashi Kokubun\)](#)

FYI: I added workaround to load fiddle from bundled gems at rjit.rb now.

<https://github.com/ruby/ruby/blob/master/rjit.rb#L33>

This workaround should be removed from rjit.rb. I will do that until final release or Ruby 3.5.

#41 - 06/18/2025 10:08 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #21442: Make tsort to bundled gems added