Ruby - Feature #18376

Version comparison API

12/01/2021 04:56 PM - vo.x (Vit Ondruch)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Is there a chance to have version comparison API? For example if Gem::Version was extracted into ::Version. This idea was triggered by this PR 1 and 2, where the Gem::Version API is used for comparing Ruby versions. While RubyGems might be available everywhere, it does not look correct to introduce dependencies on RubyGems into libraries which could run without them just fine.

Related issues:

Related to Ruby - Feature #17684: Remove `--disable-gems` from release versio...	**Assigned**	
Is duplicate of Ruby - Feature #5861: String#version_compare	**Rejected**	**01/08/2012**

History

#1 - 12/01/2021 05:08 PM - Eregon (Benoit Daloze)

I think this could be useful.
I've seen several cases like this, notably I remember https://github.com/ffi/ffi/blob/master/lib/ffi.rb.
There is a workaround there: (RUBY_ENGINE_VERSION.split('.').map(&:to_i) <=> [20, 1, 0]) >= 0 since RubyGems might not be loaded yet at that point.
That's not really pretty, so a builtin Version seems nicer.

One issue I could imagine is different version conventions and how to compare them.

#2 - 12/01/2021 05:47 PM - vo.x (Vit Ondruch)

Eregon (Benoit Daloze) wrote in #note-1:

> One issue I could imagine is different version conventions and how to compare them.

Opportunity for standardization? :) RubyGems with their version checking are doing that anyway.

#3 - 12/01/2021 05:55 PM - deivid (David Rodríguez)

My opinion is that the right place for this functionality is RubyGems and that adding/duplicating it as builtin functionality because someone might need it before rubygems is loaded, or while using --disable-gems is overkill.

#4 - 12/01/2021 08:36 PM - Eregon (Benoit Daloze)

Note that for default gems (which ffi is on jruby/truffleruby) it's also not possible to use RubyGems, so I think there is a non-trivial amount of cases.

Additionally, if people use bundler standalone mode, there is also no RubyGems loaded (or it's suboptimal to load it).

#5 - 12/01/2021 09:24 PM - deivid (David Rodríguez)

Yeah, I still think it's completely overkill, and people will be confused because we will have two different things that do the same thing.

#6 - 12/01/2021 10:59 PM - vo.x (Vit Ondruch)

RubyGems could adopt this API and slowly phase out its own version.

#7 - 12/02/2021 07:47 AM - pocke (Masataka Kuwabara)

::Version will be conflict with optparse's convention. Probably we need to consider the naming.
ref: https://github.com/ruby/ruby/blob/fe506d7945788f4c3243e9ec25c20c5dbd315073/lib/optparse.rb#L1208-L1213

#8 - 12/02/2021 08:10 AM - naruse (Yui NARUSE)

- Related to Feature #17684: Remove `--disable-gems` from release version of Ruby added

#9 - 12/02/2021 08:11 AM - naruse (Yui NARUSE)

- Is duplicate of Feature #5861: String#version_compare added

#10 - 12/02/2021 08:16 AM - naruse (Yui NARUSE)

If you are saying we want to use some rubygems feature with --disable-gems, we need to raise #17684 again.

#11 - 12/02/2021 09:56 AM - deivid (David Rodríguez)

> RubyGems could adopt this API and slowly phase out its own version.

Thanks for bringing that up. The more I think about it, the more I dislike it. I don't think version comparison is common enough to be something builtin to core, let alone if it's only to satisfy the needs of a handful of default gems (only ffi has been mentioned so far). It feels to me that in order to save one line of code in ffi, I have to go through the trouble of slowly phasing out what's probably the most commonly used rubygems constant in the wild. Definitely not a fan :sweat_smile:.

#12 - 12/02/2021 10:35 AM - Eregon (Benoit Daloze)

String#version_compare (#5861) would likely be a much less invasive way to add this, and so probably better.

#13 - 12/02/2021 10:45 AM - deivid (David Rodríguez)

That feature request was rejected by Matz because of not being common enough, and nothing seems to have changed in that regard.

#14 - 12/02/2021 03:28 PM - austin (Austin Ziegler)

deivid (David Rodríguez) wrote in #note-11:

> > RubyGems could adopt this API and slowly phase out its own version.
>
> Thanks for bringing that up. The more I think about it, the more I dislike it. I don't think version comparison is common enough to be something builtin to core, let alone if it's only to satisfy the needs of a handful of default gems (only ffi has been mentioned so far). It feels to me that in order to save one line of code in ffi, I have to go through the trouble of slowly phasing out what's probably the most commonly used rubygems constant in the wild. Definitely not a fan :sweat_smile:.

I have at least two libraries that I maintain using the Gem::Version API for feature enablement or back-port support. Some libraries that I have used for projects set VERSION = Gem::Version.new('1.2.3'); at least some of the gems that I have used in the past *also* use the version API in the same way that I do.

People reach for Gem::Version because it's the most feature-complete readily available semver library for Ruby. As a comparison, Elixir *does* provide a Version module as part of core.

If we don't want to add a ::Version constant to core, then perhaps the *best* thing to do would be for Gem::Version to be made available *separately* from the rest of RubyGems, possibly importing it into core (this has the benefit of *not* requiring an API change for anything that *uses* it). Or maybe (and this is a bit of a bootstrap problem), Gem::Version could be packaged as its own gem for people to import separately if they need gem versioning.

This isn't a *frequent* problem, but I disagree that it's "not common enough". I have *no clue* what would happen with the gems that I maintain (mime-types is perhaps the easiest version to test) if run with --disable-gems. I think that this is worth adding to core.

#15 - 12/02/2021 04:11 PM - deivid (David Rodríguez)

I'm definitely aware that this is used for what you point out. But most gem authors just use it and don't worry about "what if rubygems is not available?", which I think it's reasonable because rubygems is part of ruby and they are developing a gem after all.

#16 - 12/02/2021 04:32 PM - vo.x (Vit Ondruch)

There is probably more to this issue.

1. On the first place, I'd love if everybody used introspection instead of version checks. We would not need to have this conversation. But honestly, I don't really know how to detect the kwargs.
2. Ruby used to have simple version comparison up until 2.1.10. The string comparison worked just fine. Nevertheless "-preview1" would break it anyway.
3. I don't know what is suggested way to compare Ruby versions since Ruby 2.1.10. String#version_compare would be certainly quite easy interface. Or RUBY_VERSION could provide some operator for version comparisons. Gem::Version.new("2.6.18")<=>Gem::Version.new("2.6.3") was pointed out already in #5861 but that means the third party dependency.
4. If the third party dependency is deemed OK, then it should be accompanied by require "rubygems/version" to ensure it is available. But that on itself won't work and require "rubygems" is needed. I don't think that it can be assumed that dependencies are loaded.

#17 - 12/02/2021 04:42 PM - vo.x (Vit Ondruch)

deivid (David Rodríguez) wrote in #note-15:

> But most gem authors

This is OT, but just FTR, I might be considered "gem author", but I am Ruby developer on the first place. RubyGems are mostly convenient way of code distribution, but there are other ways (e.g. git clone or RPM/DNF on my platform).

#18 - 12/02/2021 04:50 PM - austin (Austin Ziegler)

deivid (David Rodríguez) wrote in #note-15:

> I'm definitely aware that this is used for what you point out. But most gem authors just use it and don't worry about "what if rubygems is not available?", which I think it's reasonable because rubygems is part of ruby and they are developing a gem after all.

Yet there's people who use --disable-gems for various reasons, yet might also use a standalone bundle. I just tried this and I *do* get an uninitialized constant error with --disable-gems. I know that gem authors treat Rubygems as a given, but it feels…iffy to me that we depend on this this way. I did just see that Gem::Version does not get defined in an idempotent way, as it is declared with class Gem::Version and not class Gem; class Version: ruby --disable-gems -rrubygems/version -e 'p Gem::Version' results in uninitialized constant Gem.

At any rate, I am using Elixir much more often these days, where I *do* have a Version module built-in for those (admittedly rare) cases where I need version comparison—and it is *really* nice to have something readily available as part of the core.

#19 - 12/02/2021 04:54 PM - deivid (David Rodríguez)

Of course, but I believe most people who need this are gem authors, and most gem authors use rubygems. People developing apps usually support a single version of gems or ruby, so generally don't need this kind of feature? I think most Ruby developers assume they can use Gem::Version freely, because Rubygems is normally available by default.

Anyways, I think my opinion is pretty clear so I'll refrain from commenting further.

#20 - 12/02/2021 04:56 PM - deivid (David Rodríguez)

Sorry @Austin, I was actually replying to @vo.x in my previous comment, but our comments crossed. Anyways, I don't have much to add, really!

#21 - 12/03/2021 03:56 AM - unasuke (Yusuke Nakamura)

I think it is useful for not just gem authors but also application developers.
In my known case of providing an SDK and API as service, version string compare is often used in parsing SDK version string on request from SDK to API server.
Gem::Version is commonly used for comparing "semantic version"-ed version string in that situation. Because it can use without any other gem installed. But that is not "Gem". It brings tiny context confusion.

If I can use the same function with ::Version, it's useful. Not that I'm against it, but I am not sure if it's worth introducing this as a standard library because it might require a lot of work to resolve all of the above concerns.

#22 - 12/06/2021 09:02 AM - deivid (David Rodríguez)

Thinking a bit more about this, current Gem::Version functionality seems quite independent, so there may be a way to provide this feature that's also low friction for rubygems maintainers.

Something like https://github.com/rubygems/rubygems/pull/5136 would already address the "I don't want to load all of rubygems just to compare some versions" concern. But if we extracted rubygems version.rb file to a new default gem, that would provide the same functionality through the ::Version class for users that explicitly require "version", also addressing the "Gem::Version is confusing for some version usages not related to gems" concern. Nothing would have to change for us in rubygems, we would need to vendor version.rb under the Gem namespace like we do for other default gems, which is exactly what we are already doing.

#23 - 12/06/2021 12:36 PM - Eregon (Benoit Daloze)

deivid (David Rodríguez) wrote in #note-22:

> Something like https://github.com/rubygems/rubygems/pull/5136 would already address the "I don't want to load all of rubygems just to compare some versions" concern.

Unfortunately this does not work if RubyGems is loaded lazily via autoload :Gem, 'rubygems' (it will load all of RubyGems on require 'rubygems/version').

And also this seems quite confusing if the Gem constant is defined but all the constants/classes under Gem are not there and would be NameError when accessing them, unless require 'rubygems' is done before those accesses:

```
$ ruby --disable-gems -e 'require "rubygems/version"; p Gem::Version; p Gem::Specification'
Gem::Version
-e:1:in `<main>': uninitialized constant Gem::Specification (NameError)
```

For this reason, IMHO that PR should be reverted.

> But if we extracted rubygems version.rb file to a new default gem, that would provide the same functionality through the ::Version class for users

> that explicitly require "version", also addressing the "Gem::Version is confusing for some version usages not related to gems" concern. Nothing would have to change for us in rubygems, we would need to vendor version.rb under the Gem namespace like we do for other default gems, which is exactly what we are already doing.

That seems a great idea to me.

#24 - 12/06/2021 12:49 PM - deivid (David Rodríguez)

> And also this seems quite confusing if the Gem constant is defined but all the constants/classes under Gem are not there and would be NameError when accessing them, unless require 'rubygems' is done before those accesses:

I mean, of course, right? What else would you expect? Isn't that the whole point?

#25 - 12/06/2021 12:52 PM - deivid (David Rodríguez)

To clarify, won't users of --disable-gems will expect that nothing under Gem is defined except for the explicitly required rubygems/version? Isn't that what they want?

#26 - 12/06/2021 12:56 PM - Eregon (Benoit Daloze)

deivid (David Rodríguez) wrote in #note-24:

> I mean, of course, right? What else would you expect? Isn't that the whole point?

Not really, I think the whole point is to have a version API without needing to load RubyGems and working even if RubyGems is not loaded (--disable-gems, Bundler standalone, etc).
The side-effect of breaking any access under Gem in some cases is IMHO bad enough that that PR is not worth it.
It'll break any defined?(Gem) usage, etc.
We've had empty YAML, Date and other modules defined in the past, it's really confusing and IMHO should be avoided at all costs.

The version default gem idea OTOH has none of these problems, so I think that's the way to go for this issue.

#27 - 12/06/2021 01:16 PM - deivid (David Rodríguez)

I disagree that this will "break" anything, but I'll revert for now until we gather some more opinions on my better idea.

#28 - 12/06/2021 08:30 PM - deivid (David Rodríguez)

We reverted it for now.

#29 - 12/30/2021 10:33 AM - janosch-x (Janosch Müller)

pocke (Masataka Kuwabara) wrote in #note-7:

> ::Version will be conflict with optparse's convention. Probably we need to consider the naming.

Maybe Semver?

This is a bit clunky but would arguably indicate the core functionality better than Version. (Or SemVer, but lower case "v" would match Errno, Regexp.)

#30 - 04/02/2024 06:34 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #19744: Namespace on read added

#31 - 04/02/2024 06:35 AM - hsbt (Hiroshi SHIBATA)

- Related to deleted (Feature #19744: Namespace on read)