

Ruby master - Feature #5481

Gemifying Ruby standard library

10/25/2011 06:44 AM - nahi (Hiroshi Nakamura)

Status:	Assigned
Priority:	Normal
Assignee:	hsbt (Hiroshi SHIBATA)
Target version:	
Description	
=begin	
Up-to-date summary of this proposal is at ((URL:https://bugs.ruby-lang.org/projects/ruby/wiki/StdlibGem))	
== Motivation	
<ul style="list-style-type: none">• ruby's release cycle is slow for some standard libraries;<ul style="list-style-type: none">◦ ex. security fix for WEBrick, xmlrpc and Zlib.◦ ex. API iteration for net/http, OpenSSL, json, psych, RDoc, minitest and rake.• There's already the feature called 'default gems' in ruby and some stdlibs are already using it:<ul style="list-style-type: none">◦ rake, rdoc, minitest, json, io-console, bigdecimal◦ And some gems are already doing out-of-band releases.• When releasing we should give independence equally to all stdlibs, but in a consistent and controllable way.	
== Proposal	
<ul style="list-style-type: none">• Allow out-of-band stdlib releases.<ul style="list-style-type: none">◦ We are not proposing changes to ruby's release management, the release manager would decide when they release ruby and stdlib.• Allow more stdlibs to be installed as a 'default gem'• Register these gems on RubyGems.org<ul style="list-style-type: none">◦ Introduce a new mechanism: controlling supported ruby version so that we can avoid installing unexpected version of stdlib gems. For example, a WEBrick gem for ruby 2.0.1 (released from ruby_2_0_1 branch) should not be installed for ruby 2.0.0 (released from ruby_2_0_0 branch) unless we know it works for both 2.0.0 and 2.0.1.	
Note:	
<ul style="list-style-type: none">• Moving stdlibs repository location is not a target of this proposal. The implementation details of stdlib gems should hide this from ruby committers.• ruby_1_9_3 is not a target of this proposal. The change should be introduced from 2.0.0 release.	
...Some more details of the proposal and discussion topics are going to follow as comments.	
=end	
Related issues:	
Related to Ruby master - Feature #9612: Gemify OpenSSL	Closed
Related to Ruby master - Bug #10610: "make install" fails without zlib	Closed 12/17/2014
Related to Ruby master - Feature #11025: Gemify Rake	Closed 04/02/2015
Related to Ruby master - Feature #11057: Gemify JSON	Rejected
Related to Ruby master - Feature #12160: Extract XMLRPC library to bundled gem	Closed
Related to Ruby master - Feature #8526: gemify tk	Closed
Related to Ruby master - Misc #13072: Current state of date standard library	Open
Related to Ruby master - Feature #13173: Gemify webrick	Closed
Related to Ruby master - Feature #13177: Gemify csv	Closed
Related to Ruby master - Feature #13182: Gemify cmath	Closed
Related to Ruby master - Feature #13183: Gemify date	Closed
Related to Ruby master - Feature #13197: Gemify fileutils	Closed
Related to Ruby master - Feature #13201: Gemify dbm	Closed
Related to Ruby master - Feature #13199: Gemify strscan	Closed
Related to Ruby master - Feature #13186: Gemify zlib	Closed

Related to Ruby master - Feature #11083: Gemify net-telnet	Closed	
Related to Ruby master - Feature #8539: Unbundle ext/tk	Closed	
Related to Ruby master - Bug #6123: Properly gemify BigDecimal	Rejected	03/07/2012
Related to Ruby master - Feature #2635: Unbundle rdoc	Rejected	01/23/2010
Related to Ruby master - Feature #13213: Gemify scanf	Closed	
Related to Ruby master - Feature #13248: Gemify gdbm	Closed	
Related to Ruby master - Feature #13261: Gemify sdbm	Closed	
Related to Ruby master - Feature #9456: Include bin/racc with ruby	Closed	
Related to Ruby master - Misc #13771: Digest, Ruby OpenSSL, OpenSSL v1.1.0	Closed	
Related to Ruby master - Feature #16170: Remove the unmaintained libraries fr...	Closed	
Related to Ruby master - Feature #15657: Make webrick to bundled gems	Assigned	
Related to Ruby master - Misc #16778: Should we stop vendoring default gems c...	Rejected	

History

#1 - 10/25/2011 06:49 AM - nahi (Hiroshi Nakamura)

- =begin
 == Implementation
- === Install stdlibs as 'default gems'
- Newly created stdlib gems version scheme: ruby's version + '.n'(dot plus a number)
 - ex. WEBrick 'default gem': 2.0.0.0, 2.0.0.1, ...
 - Gems which supports multiple ruby versions (ex. rake, rdoc, etc.) keeps their version.
 - Controlling supported ruby version by specifying "platform". In RubyGems, currently the ruby platform only allows a major and minor version (1.8 or 1.9)
 - Uninstalling 'default gems' should be blocked to avoid confusion. RubyGems bundled with 1.9.3 allows users to uninstall 'default gems' now. % gem uninstall webrick -v 2.0.0.0 #=> should raise an error.
 - An updated version of an stdlib gem should be treated as regular gems. With the current 'default gems' implementation, a newer gem will not automatically override the default gem with 'require': % ruby -rwebrick -e 'p WEBrick::VERSION' #=> 2.0.0.0 % gem update webrick Updating installed gems Updating webrick Fetching: webrick-2.0.1.5.gem (100%) Successfully installed webrick-2.0.1.5 Gems updated: webrick Installing ri documentation for webrick-2.0.1.5... Installing RDoc documentation for webrick-2.0.1.5... % ruby -rwebrick -e 'p WEBrick::VERSION' #=> 2.0.0.0 << should be 2.0.1.5
 - Make autoload for stdlib gems work as long as autoload feature exists in 2.0. % ruby -e 'autoload :JSON, "json"; p JSON' #=> "JSON"

For existing 'default gems' see tool/rbinstall.rb. It installs stdlibs as 'default gems'.

- It scans {lib,ext}/*/*.gemspec and installs it as a spec-only gems via rubygems.
- Then installs gemdir/bin/* as executables (it would be better to use rubygems instead.)
- It also reads defs/default_gems, but those are obsolete.

=== Register 'default gems' on RubyGems.org

- Improve 'default gems' for creating gem from stdlib file/directory layout.
- stdlib gems maintainer registers the updated gems on RubyGems.org.
 - Maintainer must have an account at RubyGems.org

== ToDo

- After installing updated stdlib gems, those should be treated as regular gems. How?
 - Installing 'default gems' as a real gem to /usr/local/lib/ruby/default_gems/, not to /usr/local/lib/ruby/site_ruby/
 - advantage: simple implementation
 - drawback: stdlib gems cannot be loaded with --disable-gems.
 - Implement it as a new feature of RubyGems.
 - advantage: stdlib can be loaded without --disable-gems
 - drawback: may penalize ruby startup time
 - drawback: may be complex to implement

The RubyGems team tried to implement a feature similar to this for ruby 1.9.1 and ruby 1.9.2, but it did not work out very well...
 (3) ?

- To avoid installing an incompatible version of stdlib gems, update RubyGems to allow three digits (1.8/1.9 -> 1.9.X) on a "ruby" platform in the gem spec.
- Uninstalling 'default gems' should be blocked. Set default path to /usr/local/lib/ruby/default_gems/
- Improve 'default gems' for creating gem from stdlib file/directory layout.
- Decide which stdlibs should be gemified. Let's ask maintainers. We would need to find maintainers of some stdlibs if needed.

- Decide tagging/branching policy for stdlib gems. Up to the release manager?
- Find a way to allow autoloading stdlib gems.
 - It's because autoload does not respect require overwriting now. When we install 'default gems' as a real gem, it needs some care to work.

== What stdlibs should be gemified?

=== Principle

- The lib must have a maintainer.
- The lib should be highly independent from ruby's code base.
 - Feature dependencies are OK since RubyGems can resolve them. ex. net/http -> openssl

=== Stdlib status

- lib/benchmark
- lib/cgi
- lib/csv
- lib/drb
- lib/erb
- lib/irb
- lib/logger: NaHi will maintain this as a 'default gem'
- lib/optparse
- lib/pstore
- lib/racc
- lib/rexml
- lib/rinda
- lib/rss
- lib/webrick: NaHi will maintain this as a 'default gem'
- lib/xmlrpc
- lib/yaml
- lib/rake
- lib/net: Do we split this to core, ftp, http (and https), mail (imap, pop and smtp) and telnet?
- ext/curses
- ext/date
- ext/digest
- ext/iconv: Deprecated. New products and libraries should not use this
- ext/openssl
- ext/sych

Already a 'default gem':

- lib/minitest
- lib/rake
- lib/rdoc
- lib/rubygems
- ext/bigdecimal: Not yet registered at RubyGems.org (Mrkn will register that to RubyGems.org after RubyConf 2011).
- ext/io-console: Not yet registered at RubyGems.org
- ext/json
- ext/psych

== Call for discussion

- Topics at ToDo above.
- What stdlibs should be gemified?

Please post comments to this ticket. I'll update the Wiki page at <https://redmine.ruby-lang.org/projects/ruby/wiki/StdlibGem> as a summary.
=end

#2 - 10/25/2011 06:51 AM - nahi (Hiroshi Nakamura)

=begin

== Background

There was an awesome keynote by Aaron at RubyKaigi2011.

- RubyKaigi 2011 keynote by Aaron Patterson (tenderlove) / AT&T Interactive: (([URL:http://vimeo.com/26507951](http://vimeo.com/26507951)))
- Most relevant parts for this topic
 - 14'27" -- 25'55": Ruby's release cycle and gem inconsistencies
 - 24'05" -- 27'25": Converting stdlib into gems and release them along with the ruby tarball ("That's one thing I'd like to see changed")

At the next session, Matz and Japanese ruby-core committers discussed the next ruby release. Matz said 'there will be no 1.9.4 because it becomes 2.0'. (Note: See the discussion at (([URL:https://redmine.ruby-lang.org/issues/5056](https://redmine.ruby-lang.org/issues/5056))).)

Any detailed changes for 2.0 were not known but I (NaHi) thought that gemifying stdlib should happen before the changes because it would include incompatible behaviors. The work for gemifying would be postponed after 2.0 release (we must be busy for stabilizing 2.0 release.) I was thinking that gemifying stdlib for faster iteration (shorter release cycle) which had been explained in Aaron's keynote was a must and I wanted to see that happen soon. I talked to Aaron(tenderlove) and Eric(drbrain, a maintainer of RubyGems) first about how we could make it happen and to make this proposal.

I also asked Matz about it.

- (([URL:https://twitter.com/#/nahi/status/92386241156349952](https://twitter.com/#/nahi/status/92386241156349952)))

Then I got the reponse 'Talk to Yuki Sonoda(yugui) and Aaron. I say yes'.

- (([URL:https://twitter.com/#/yukihiro_matz/status/92400266493702144](https://twitter.com/#/yukihiro_matz/status/92400266493702144)))

So I talked with many ruby committers including Yuki and Aaron at RubyKaigi2011. Lots of discussion about the benefits and what must not happen. Not a unanimous accord, but almost all committers there agreed/understood what we're planning to do by gemifying stdlib.

Now, at the time 1.9.3 is being released, I want to start the work.
=end

#3 - 10/25/2011 01:23 PM - nahi (Hiroshi Nakamura)

Hi,

On Tue, Oct 25, 2011 at 12:49, Yusuke Endoh mame@tsg.ne.jp wrote:

Could you summarize advantages (if any) and disadvantages (or additional work) from POV of each person?

Sure, I will. But please don't wait to post your comment now. I should be still missing some issues like 'autoload'. I'll try to summarize per-role advantages/disadvantages once I collected initial responses.

// NaHi

#4 - 10/25/2011 01:53 PM - drbrain (Eric Hodel)

On Oct 24, 2011, at 8:49 PM, Yusuke Endoh wrote:

Hello,

Could you summarize advantages (if any) and disadvantages (or additional work) from POV of each person?

I will describe my thoughts

- user

Users can obtain improvements to parts of ruby without waiting for a new release.

No need to gem 'rdoc', gem 'psych', etc. to use the latest installed version of a gem.

With the current implementation default gems are special. This proposal will make them stop being special.

- default gem developer

Default gem developers won't have to educate users to do something special to use the latest version of the library.

Default gem developers may need to support multiple versions of ruby in their gem or backport one fix to multiple releases (like backport requests to Ruby 1.9.3 and Ruby 1.9.2).

Default gem developers can release improvements between ruby release cycles.

- rubygems developer

We may need to implement better RubyGems platform support, but the idea is applicable beyond just improving stdlib gems.

- rubygems.org admin

The rubygems.org admin should not need to do anything special.

#5 - 10/25/2011 02:57 PM - lucas (Lucas Nussbaum)

(With my Debian hat)

I don't really like the plan of having releases of libraries both with the interpreter and as separate gems. Duplicating distribution paths for libraries will make things very complex since there would be no single "latest version" of a given library. We already have that problem with rubygems due to its bundling in 1.9.

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

One benefit that this would bring is that interpreter implementations could all use the same stdlib. Currently jruby and rubinius are both lagging behind in that regard. This would reinforce a healthy competition between implementations, and make shipping several implems easier to support.

#6 - 10/25/2011 09:53 PM - trans (Thomas Sawyer)

On Oct 25, 1:58 am, Lucas Nussbaum lu...@lucas-nussbaum.net wrote:

Issue [#5481](#) has been updated by Lucas Nussbaum.

(With my Debian hat)

I don't really like the plan of having releases of libraries both with the interpreter and as separate gems. Duplicating distribution paths for libraries will make things very complex since there would be no single "latest version" of a given library. We already have that problem with rubygems due to its bundling in 1.9.

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

One benefit that this would bring is that interpreter implementations could all use the same stdlib. Currently jruby and rubinius are both lagging behind in that regard. This would reinforce a healthy competition between implementations, and make shipping several implems easier to support.

+1.

I mean, how hard is it to add gem install stdlib to end of make install routine? Where stdlib would be an (otherwise empty) meta-package with dependencies to all standard gems.

#7 - 10/25/2011 09:59 PM - judofyr (Magnus Holm)

On Tue, Oct 25, 2011 at 14:45, Intransion transfire@gmail.com wrote:

On Oct 25, 1:58 am, Lucas Nussbaum lu...@lucas-nussbaum.net wrote:

Issue [#5481](#) has been updated by Lucas Nussbaum.

(With my Debian hat)

I don't really like the plan of having releases of libraries both with the interpreter and as separate gems. Duplicating distribution paths for libraries will make things very complex since there would be no single "latest version" of a given library. We already have that problem with rubygems due to its bundling in 1.9.

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

One benefit that this would bring is that interpreter implementations could all use the same stdlib. Currently jruby and rubinius are both lagging behind in that regard. This would reinforce a healthy competition between implementations, and make shipping several implems easier to support.

+1.

I mean, how hard is it to add gem install stdlib to end of make install routine? Where stdlib would be an (otherwise empty) meta-package with dependencies to all standard gems.

That would require internet connection. I think you should be able to install a complete version of Ruby without an internet connection.

So we still have to ship the gems with the release, which is what Lucas doesn't like...

#8 - 10/25/2011 10:23 PM - mame (Yusuke Endoh)

Hello,

2011/10/25 Hiroshi Nakamura nakahiro@gmail.com:

Sure, I will. But please don't wait to post your comment now. I should be still missing some issues like 'autoload'. I'll try to summarize per-role advantages/disadvantages once I collected initial responses.

Who will upload stdlib gems to rubygems.org? The release manager?
Each stdlib gem developer?

When a security issue of stdlib gems is reported to the security team (security@ruby-lang.org), how should it be handled?
If a new tarball of ruby distribution and a new gem file is both released, the release (and announce) time lag must be minimized.

What is the version policy of stdlib gems? Any new feature or even incompatibility may be included in the forth digit update?
I'm concerned about a relationship with the policy of patch release.

Is no factor that prevent us from releasing Ruby increased, even when stdlib gem developer is too busy to work for Ruby?

Small questions to your proposal follows:

2011/10/25 Hiroshi Nakamura nakahiro@gmail.com:

== Motivation

- When releasing we should give independence equally to all stdlibs, but in a consistent and controllable way.

I cannot understand this point. Could you erabolate?

== Proposal

Note:

- Moving stdlibs repository location is not a target of this proposal. The implementation details of stdlib gems should hide this from ruby committers.

I cannot understand the context of the second sentence.
What do you mean?

== ToDo

NaHi-san will do (or coordinate) all the Todos, right? :-)

- Find a way to allow autoloading stdlib gems.
 - It's because autoload does not respect require overwriting now. When we install 'default gems' as a real gem, it needs some care to work.

I recommend you create another ticket about autoload, with a focus on what feature you need in the ruby core.

--

Yusuke Endoh mame@tsg.ne.jp

#9 - 10/25/2011 10:23 PM - mame (Yusuke Endoh)

Hello, Eric

Thank you. Your explanation helped me understand.

--

Yusuke Endoh mame@tsg.ne.jp

#10 - 10/25/2011 10:53 PM - mame (Yusuke Endoh)

Hello,

2011/10/25 Lucas Nussbaum lucas@lucas-nussbaum.net:

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

I cannot understand your issue.
Even if we release core+stdlib as a big package, you can split the package to each deb package. Doesn't it work?

--
Yusuke Endoh mame@tsg.ne.jp

#11 - 10/26/2011 12:23 AM - trans (Thomas Sawyer)

On Tue, Oct 25, 2011 at 8:56 AM, Magnus Holm judofyr@gmail.com wrote:

On Tue, Oct 25, 2011 at 14:45, Intransion transfire@gmail.com wrote:

On Oct 25, 1:58 am, Lucas Nussbaum lu...@lucas-nussbaum.net wrote:

Issue [#5481](#) has been updated by Lucas Nussbaum.

(With my Debian hat)

I don't really like the plan of having releases of libraries both with the interpreter and as separate gems. Duplicating distribution paths for libraries will make things very complex since there would be no single "latest version" of a given library. We already have that problem with rubygems due to its bundling in 1.9.

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

One benefit that this would bring is that interpreter implementations could all use the same stdlib. Currently jruby and rubinius are both lagging behind in that regard. This would reinforce a healthy competition between implementations, and make shipping several implems easier to support.

+1.

I mean, how hard is it to add gem install stdlib to end of make install routine? Where stdlib would be an (otherwise empty) meta-package with dependencies to all standard gems.

That would require internet connection. I think you should be able to install a complete version of Ruby without an internet connection.

I think it's time to accept that we are in an Internet world.

So we still have to ship the gems with the release, which is what Lucas doesn't like...

Moreover, it is possible to package up a set of gems into a ship-able and install-able bundle for those who really need that.

#12 - 10/26/2011 01:23 AM - lucas (Lucas Nussbaum)

On 26/10/11 at 00:13 +0900, Trans wrote:

On Tue, Oct 25, 2011 at 8:56 AM, Magnus Holm judofyr@gmail.com wrote:

On Tue, Oct 25, 2011 at 14:45, Intransion transfire@gmail.com wrote:

On Oct 25, 1:58 am, Lucas Nussbaum lu...@lucas-nussbaum.net wrote:

Issue [#5481](#) has been updated by Lucas Nussbaum.

(With my Debian hat)

I don't really like the plan of having releases of libraries both with the interpreter and as separate gems. Duplicating distribution paths for libraries will make things very complex since there would be no single "latest version" of a given library. We already have that problem with rubygems due to its bundling in 1.9.

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

One benefit that this would bring is that interpreter implementations could all use the same stdlib. Currently jruby and rubinius are both lagging behind in that regard. This would reinforce an healthy competition between implementations, and make shipping several implems easier to support.

+1.

I mean, how hard is it to add gem install stdlib to end of make install routine? Where stdlib would be an (otherwise empty) meta-package with dependencies to all standard gems.

That would require internet connection. I think you should be able to install a complete version of Ruby without an internet connection.

I think it's time to accept that we are in an Internet world.

So we still have to ship the gems with the release, which is what Lucas doesn't like...

What I don't like with the current situation is the fact that we currently have some codebases that are forked between stdlib and gems.

In the 2.0 model, I would be completely fine if the stdlib was shipped with the interpreter(s) provided that it is only provided as a convenience snapshot. What we (Debian) would probably do is follow stdlib releases independently of interpreter releases, and just ignore the stdlib gem provided with the interpreter.

Moreover, it is possible to package up a set of gems into a ship-able and install-able bundle for those who really need that.

Exactly.

Lucas

#13 - 10/26/2011 01:23 AM - lucas (Lucas Nussbaum)

On 25/10/11 at 22:43 +0900, Yusuke Endoh wrote:

Hello,

2011/10/25 Lucas Nussbaum lucas@lucas-nussbaum.net:

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

I cannot understand your issue.

Even if we release core+stdlib as a big package, you can split the package to each deb package. Doesn't it work?

Let me try to rephrase more clearly.

There are several places different sets of software that are or could be developed independently in the Ruby ecosystem.

- interpreters (+ core libs)
- standard library
- third-party gems

The problem is: how do we maintain and release interpreters and standard library? There are two sides to this question:

- how do we avoid stdlib code duplication between interpreters?
- how do we encourage stdlib development?

I think that the cleanest solution is to consider MRI and the stdlib to be two completely different projects, with their own release cycles, their own security support, etc. That way:

- people working on the interpreters can focus on the interpreters
- people working on the stdlib can focus on the stdlib
- there's no code duplication

There are two problems with that solution:

- it requires each stdlib release to be well tested on the supported interpreters, possibly taking into account past releases of the interpreters.
- there could be a problem during the installation. We need to bootstrap a ruby installation so that there's enough of stdlib installed to run rubygems and install the remaining gems. That can easily be solved by providing a full snapshot of stdlib. But then it needs to stay a snapshot, and everybody should be expected to be able to use a newer version of each lib.

With the current proposal, I see two ways for Debian to package Ruby (and none of them are really good):

- provide a bundle of stdlib + mix of updated gems as a single package. That means that Debian maintainers will have to choose which version to include for each gem, and the user won't be able to do it.
- split the stdlib into a very large number of small packages for each lib, so that each one can be updated separately. A prerequisite for this solution is that we know beforehand how to split the stdlib. (The root problem with Debian is that two packages cannot provide the same file if they can be installed together. So we can't have a package replacing/overriding only a part of another package.)

Lucas

#14 - 10/26/2011 05:23 AM - sdaubert (Sylvain Daubert)

Le 25/10/2011 17:13, Trans a écrit :

On Tue, Oct 25, 2011 at 8:56 AM, Magnus Holm judofyr@gmail.com wrote:

On Tue, Oct 25, 2011 at 14:45, Intransition transfire@gmail.com wrote:

On Oct 25, 1:58 am, Lucas Nussbaum lu...@lucas-nussbaum.net wrote:

Issue [#5481](#) has been updated by Lucas Nussbaum.

(With my Debian hat)

I don't really like the plan of having releases of libraries both with the interpreter and as separate gems. Duplicating distribution paths for libraries will make things very complex since there would be no single "latest version" of a given library. We already have that problem with rubygems due to its bundling in 1.9.

Instead, I would prefer a plan where the interpreter no longer releases with the stdlib, but just releases with a "core lib". Then the user would install the stdlib as a set of gems.

One benefit that this would bring is that interpreter implementations could all use the same stdlib. Currently jruby and rubinius are both lagging behind in that regard. This would reinforce a healthy competition between implementations, and make shipping several implems easier to support.

+1.

I mean, how hard is it to add gem install stdlib to end of make install routine? Where stdlib would be an (otherwise empty) meta-package with dependencies to all standard gems.

That would require internet connection. I think you should be able to install a complete version of Ruby without an internet connection.

I think it's time to accept that we are in an Internet world.

Hi,

Sometimes, ruby must be installed on systems who may not be connected to internet (for security reasons for example).

Sylvain

#15 - 10/26/2011 07:23 AM - drbrain (Eric Hodel)

On Oct 25, 2011, at 5:56 AM, Magnus Holm wrote:

That would require internet connection. I think you should be able to install a complete version of Ruby without an internet connection.

The current proposal does not envision installing gems from rubygems.org or any other non-local source when you run `make install`.

You won't need an internet connection to build and install ruby, just the tarball. The source layout is exactly as it is now and gems are created at install time on your machine from the tarball through a script in `tool/`.

(At release time these gems are also uploaded to rubygems.org.)

#16 - 10/26/2011 07:23 AM - drbrain (Eric Hodel)

On Oct 25, 2011, at 8:13 AM, Trans wrote:

I think it's time to accept that we are in an Internet world.

But we aren't.

There are many users of Ruby and RubyGems that can't access the internet from the computers they are using ruby on. The only way they can install ruby and gems on their machines is via a "sneaker net" of USB keys.

See:

<http://help.rubygems.org/kb/rubygems/installing-gems-with-no-network>

#17 - 10/26/2011 09:52 AM - MartinBosslet (Martin Bosslet)

I really like the proposal. Just wanted to add one thought - if we add yet more power to the gem infrastructure, we should probably revisit the idea of offering code-signed gems. The theory exists [1], we could build on that. This might be slightly off-topic, so I apologize in advance, but I still feel it is relevant, especially when gemifying stdlib, please let me explain why.

If the stdlib gets gemified we will have even more gem downloads than today. I'm not talking about the bundled gems that would be accessed locally when building Ruby for the first time (wrt to this proposal), rather about "updating" and adding gems to an existing installation. Updating will be a crucial task in production - people would certainly show interest in being able to stay up to date as fast as possible regarding security fixes, performance improvements etc.

Currently, these gem downloads are entirely unprotected, so just imagine a production deployment that connects to the outside world using a company proxy. The guy operating the company proxy could have a lot of fun serving "customized" gems that do all sorts of evil - nobody would notice. Serving the gems over a non-secure channel is definitely a risk. This might not seem such a big deal at first glance, but just think about discussing your architecture with a bank or other high security environments. Those people constantly give us the "<>" talk, so I could very well imagine that this is an issue when it comes to adopting Ruby in these environments. These people are often very conservative in their views, and even if they are still coding VB 6 GUIs and have no idea what they are talking about, still (in my own experience) they will readily accept *signed* Java applets... That's why I think offering signed gems would be helpful in sending a signal to those "enterprisys" clients, if not only to take away their arguments.

What we often find in these situations are SHA-1 checksums or similar features published on the download site. If the site is served over http this has effectively *zero* additional security. An attacker would simply man-in-the-middle the HTML of the site, too, handing you a different hash. In addition, even when sent over https, let's face it, nobody really compares these hashes manually.

I see two relatively simple options for mitigation: either secure the transport channel on the transport layer using TLS or secure the payload itself and therefore sign the gem. I prefer the second solution because it "persists" the security of the gem, the signature is tightly coupled with the gem itself and persists once downloaded, so that for example local gem cache servers would still be an option (they wouldn't be in the TLS case, unless effectively re-signing the gems).

Please note that a code signature will of course not guarantee that the code is free of bugs or not malicious. The only security it will provide is that one can be sure that the gem itself has not been modified after the holder of the code signing certificate applied the signature. Not more, not less. This narrows your trust decision down to asking yourself the question whether you want to trust the certificate holder to have validated the code to do what it is supposed to do. The person signing the gem is the last person who was able to alter the code. After applying the signature, the gem is effectively sealed. So you would have to base your trust on two things. Whether you trust this person to not have altered the code in malicious ways and whether you trust that person to have validated the code to ensure its proper functionality. Although far from perfect, in my eyes this is still a major improvement over having no guarantee at all.

A problem with code signing gems is that nobody is keen on buying a "real" code signing certificate to sign their code, but on the other hand nobody would be keen on trusting a self-signed certificate either. This is probably one of the major issues why code signing is a rarely used feature. Now since gems are stored in a centralized repository, a solution could be to not require the *authors* to sign a gem but to sign a gem on the RubyGems server immediately after uploading, using a certificate that is exclusively issued to RubyGems. Solving the problem of trusting one dedicated certificate is easy, there are several options that would still need to be discussed, but it is certainly doable. However, this would put the burden of managing this certificate and securing access to it on the shoulders of the RubyGems maintainers.

I would also suggest to keep this entirely optional. To be honest, as with all things security, this will possibly affect usability, additional configuration will likely be necessary to use the feature. So you will have to assess your individual risks - you might very well come to the conclusion that a feature like this is overkill in your individual situation. You should of course still have the option to ignore this entirely, much like today.

I talked briefly about the idea with Eric Hodel and Hiroshi Nakamura. Eric told me that similar ideas in this direction exist. I would really appreciate to hear opinions on this. If you feel that this would be indeed a good option, I would suggest to open a new issue for further discussion.

[1] <http://docs.rubygems.org/read/chapter/21>

#18 - 11/01/2011 07:05 AM - vo.x (Vit Ondruch)

Hiroshi Nakamura wrote:

- Uninstalling 'default gems' should be blocked to avoid confusion. RubyGems bundled with 1.9.3 allows users to uninstall 'default gems' now. % gem uninstall webrick -v 2.0.0.0 #=> should raise an error.

While I like your proposal, I really don't see point, why it should be disabled to uninstall the "default" gem. There might be "gem install default" command which restores the original gem version for example, but disabling uninstallation really doesn't make sense to me.

#19 - 11/01/2011 08:09 AM - drbrain (Eric Hodel)

=begin

While I like your proposal, I really don't see point, why it should be disabled to uninstall the "default" gem. There might be "gem install default" command which restores the original gem version for example, but disabling uninstallation really doesn't make sense to me.

I don't think it needs to be absolutely disabled, but I do want to prevent foot-shooting.

If you want to do extra work to uninstall default gems that should be allowed, but it should not happen without extra effort by the user.

=end

#20 - 11/02/2011 05:46 AM - vo.x (Vit Ondruch)

If you want to do extra work to uninstall default gems that should be allowed, but it should not happen without extra effort by the user.

Why do you want to make the gems special? What do you want the users to prevent from? Provide easy tool to fix the situation, not to prevent it. If somebody uninstalls default Rake for example, what is the problem to install it again?

#21 - 11/02/2011 06:24 AM - drbrain (Eric Hodel)

=begin

If net/http, fileutils, optparse, psych or zlib are released as gems uninstalling them breaks RubyGems and you will not be able to reinstall them without reinstalling ruby.

Removing such gems should be possible, I don't think it should be easy and I don't think RubyGems should go to extra effort to prevent it either.

Placing the gems in a separate path is sufficient as ((%gem uninstall%)) will only remove gems in ((GEM_HOME)) unless you use ((%-i%)).

=end

#22 - 03/08/2012 11:16 AM - nahi (Hiroshi Nakamura)

Just let it follow URL change.

#23 - 03/08/2012 04:23 PM - vo.x (Vit Ondruch)

Eric Hodel wrote:

=begin

While I like your proposal, I really don't see point, why it should be disabled to uninstall the "default" gem. There might be "gem install default" command which restores the original gem version for example, but disabling uninstallation really doesn't make sense to me.

I don't think it needs to be absolutely disabled, but I do want to prevent foot-shooting.

If you want to do extra work to uninstall default gems that should be allowed, but it should not happen without extra effort by the user.

=end

Actually if the default gems will be in some directory which is accessible using GEM_PATH and it is not current GEM_HOME, the user will not be able to uninstall the default gems. Problem solved. No new features are necessary. The only think which have to be taken in account are a /usr/bin loaders.

#24 - 03/08/2012 07:04 PM - vo.x (Vit Ondruch)

Vit Ondruch wrote:

Eric Hodel wrote:

=begin

While I like your proposal, I really don't see point, why it should be disabled to uninstall the "default" gem. There might be "gem install default" command which restores the original gem version for example, but disabling uninstallation really doesn't make sense to me.

I don't think it needs to be absolutely disabled, but I do want to prevent foot-shooting.

If you want to do extra work to uninstall default gems that should be allowed, but it should not happen without extra effort by the user.
=end

Actually if the default gems will be in some directory which is accessible using GEM_PATH and it is not current GEM_HOME, the user will not be able to uninstall the default gems. Problem solved. No new features are necessary. The only thing which have to be taken in account are a /usr/bin loaders.

And one more note. From Fedora's point of view, there is no difference between gems which originally comes with Ruby and the other Gems maintained by RPM. So it should be easy to merge the "default gems" with other "RPM managed gems". Our current operating_system.rb which sets-up the RubyGems directories can be found here [1]

[1] http://pkgs.fedoraproject.org/gitweb/?p=ruby.git;a=blob:f=operating_system.rb;hb=HEAD

#25 - 06/30/2012 01:51 PM - nahi (Hiroshi Nakamura)

- Assignee set to nahi (Hiroshi Nakamura)

#26 - 07/01/2012 12:24 AM - nahi (Hiroshi Nakamura)

- File 5481.pdf added

Endoh-san, here's "slide-show" of this proposal.

#27 - 07/01/2012 07:24 AM - nahi (Hiroshi Nakamura)

Please suggest how we can improve the slide here:

https://docs.google.com/presentation/d/15sc7sDcHRrl2v1z_Eju9KSQXhjh2WFsWFyV-MHTf7Z0/edit

#28 - 07/02/2012 02:30 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

Received, thank you.

I don't think this issue will be settled in a few minutes, but it may be valuable enough to hear matz's view.

--

Yusuke Endoh mame@tsg.ne.jp

#29 - 09/18/2012 03:23 PM - kou (Kouhei Sutou)

NaHi, could you tell us status of this work?

If you doesn't have a time to implement it, I'll implement it.

NOTE: I have another merit by this. If this feature is implemented, test-unit gem can be used without "gem 'test-unit'". So I'll implement it if nobody does it.

#30 - 09/21/2012 11:35 PM - nahi (Hiroshi Nakamura)

Kou, thanks for the cooperation. We would need your help soon.

kou (Kouhei Sutou) wrote:

NaHi, could you tell us status of this work?

I did the presentation at the last dev meeting and I think I/we need to clarify a part of the proposal for blessing from Matz. I guess Endoh-san didn't update this ticket yet? (No intention to blame. My bad, I should have asked this earlier.)

#31 - 09/22/2012 08:59 PM - kou (Kouhei Sutou)

Thanks for your reply.

I did the presentation at the last dev meeting and I think I/we need to clarify a part of the proposal for blessing from Matz.
OK. I'll wait for the answer of it.

There is another question. Could you show a list of unimplemented features. (In other words TODO list.)

Is <https://bugs.ruby-lang.org/projects/ruby/wiki/StdlibGem#ToDo> still valid? I heard another requirement from Eric. It is that "gem contents" should work for default gems. It is used by RDoc. So the list may be outdated.

#32 - 10/25/2012 02:51 PM - vo.x (Vit Ondruch)

Is the plan to extract the gems out of StdLib and make from them proper gems again? I.e. there would be for example ruby/gems folder, with rake, rdoc, minitest and later the install script would install them into lets say ruby/default_gems? There seems to be some activity [1] which is not going in this way I am afraid.

[1] <https://github.com/rubygems/rubygems/pull/377>

#33 - 10/25/2012 03:46 PM - usa (Usaku NAKAMURA)

- Target version changed from 3.0 to 2.0.0

This ticket is the center of attention at ruby-dev ML :)
NaHi-san will take a certain action in a few days, I guess.

#34 - 10/25/2012 04:26 PM - vo.x (Vit Ondruch)

What actions exactly? Will the gems be gems or just another workaround? Can I see current state? How the result will look? Can I help somehow? Thank you.

#35 - 10/25/2012 04:29 PM - vo.x (Vit Ondruch)

Will it address concerns from [#6124](#) as well?

#36 - 10/25/2012 09:01 PM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to 2.6

We discussed this request at the developers' meeting (Jul. 21), and it was not accepted.

NaHi (the proposer) presented this request by himself. He said:

Because of "fake gems", the new files of a stdlib installed by "gem update" are ignored unless a user writes gem "json" explicitly. To address this issue, stdlibs must be provided as normal gems, not "fake gems".

And the conclusion is as follows:

- The current behavior of fake gems is considered "bug", so should be fixed. But it is not a good reason to make stdlib real gems.
- If it is hard or even impossible to fix "the bug", NaHi should re-propose it.

After that, NaHi said, "fixing the behavior" effectively means "making stdlib as normal gems", but I don't know the meaning. Please wait for his words.

--

Yusuke Endoh mame@tsg.ne.jp

#37 - 10/26/2012 03:43 AM - vo.x (Vit Ondruch)

Thank you for clarification, waiting for NaHi and offering my help.

#38 - 10/30/2012 11:48 PM - vo.x (Vit Ondruch)

Hi, I'd like to question once more some parts of the proposal in wiki [1].

- After installing updated stdlib gems, those should be treated as regular gems.

The only viable solution is proposal 1. It seems that there was some work started to implement 2, but that is really wrong approach. What would be the purpose of --disable-gems then? You disable gems but they will be loaded to load some default gems? That doesn't make sense to me.

Moreover, if you have reason to use --disable-gems option, you will be probably able to use -I to setup your load paths to load the gems manually.

- Introduce a new mechanism: controlling supported ruby version

There is already mechanism in RubyGems, i.e. you can specify version of Ruby which is supported by the gem, not sure why there should be anything more. Since the StdLib gems would go into independent directory and they are uninstalleable, there can't be installed any "unexpected version of stdlib gems". If this should solve that there might be installed to new gem, then I guess Ruby developers are already used to this. So not sure what would this improve.

- Newly created stdlib gems version scheme: ruby's version + '.n'(dot plus a number)

Why WEBrick should have this kind of versioning? What if there was no change in WEBrick in between Ruby 2.0 and, 2.1? Why the version should be changed? Every gem should have independent versioning, as maintainer decides. Bind the gems versioning to Ruby versions makes no sense and it is not beneficial.

- Uninstalling 'default gems' should be blocked to avoid confusion. RubyGems bundled with 1.9.3 allows users to uninstall 'default gems' now.

Yes, since currently, the default gems are mixed with regular gems, they are uninstalleable. This would be non issue if they are in different path.

- And one more note from Fedora's Ruby maintainer point of view

It has to be possible to unbundle the default gems out of Ruby and replace them with newer versions if needed. This is similar requirement as Lucas Nussbaum pointed out earlier in this discussion. There are also different reasons, such as for example, Ruby is dependency of VIM in Fedora and VIM users really don't want to install RDoc, Minitest, Rake etc, just because they want to use VIM.

Please note that we are already doing so in Fedora and there are no issues, except [#6124](#)

[1] <https://bugs.ruby-lang.org/projects/ruby/wiki/StdlibGem>

[2] <https://github.com/rubygems/rubygems/pull/377#issuecomment-9735417>

#39 - 11/01/2012 05:36 AM - headius (Charles Nutter)

FWIW, here's how we're handling the gemification of OpenSSL...

Background: Previously, JRuby's 'openssl' support was always distributed as a gem, since we did not want to ship crypto (bouncycastle). Starting with JRuby 1.7, however, we ship 'openssl' in the standard JRuby distribution. We still want to support users on 1.6.x and want to be able to update 'openssl' independent of JRuby, so for Jruby 1.7.1 we will support using the gem if the gem is installed.

The file within JRuby is here: <https://github.com/jruby/jruby/blob/master/lib/ruby/1.9/openssl.rb>

This file attempts to load a file that's only in the gem, and the gem then adds its own dirs to LOAD_PATH so future requires will use the gem's versions of things instead of those built into JRuby.

This approach may advise a similar solution in MRI. At the very least, I'm interested in this bug so we use the same mechanism for loading the gem version of a stdlib instead of the shipped version.

I am disappointed this won't happen for 2.0.0.

#40 - 11/27/2012 11:20 PM - vo.x (Vit Ondruch)

=begin
Hi,

I played around a bit and put together a few patches. You can find it here: <https://github.com/voxik/ruby/commits/gemified-stdlib>. I took Rake as an example.

In a nutshell:

- (1) I created the gems/ subdirectory in Ruby repository.
- (2) I moved the Rake bits into gems/rake/
- (3) I added .gemspec for Rake
- (4) I modified rinstall.rb to install each gem found in gems subdirectory into standard StdLib gem location
- (5) I modified RubyGems to look into StdLib gem location for gems.

What is currently missing:

- (1) The test suite is not passing currently, but I think that the Rake's test suite should be moved into the gems/rake/ subdirectory as well and there should be some mechanism, how to execute test suites of gems.
- (2) If there would be some binary gem, its binary extension is not build ATM. However, I would say that similar approach used to build current exts could be used.

This approach has significant advantages IMO:

- (1) Rake gem currently reflect the upstream repository and as soon as the tests are moved into the gems/rake/ subfolder, it will be almost exact match. It could be easily replaced by something like svn:externals or git submodule. This would greatly improve the maintainability of the code and reduced the possibility of forking.
- (2) This would help alternative Ruby implementations to ship their additional libraries in consistent manner.

(3) Gems could be easily replaced by downstream distributions by different versions, if needed.

Any comments? Would this approach be acceptable? Should I try to spend more time polishing the test suite and binary extension's build?
=end

#41 - 11/28/2012 12:19 AM - trans (Thomas Sawyer)

Why keep the source code in the Ruby repository at all? Development occurs at the library's repository anyway, so why waste time with copying the files into Ruby repository for each release. Instead, just ship the .gem package with Ruby which can be installed via RubyGems (via rbininstall.rb) per the normal means of installing a gem.

#42 - 11/28/2012 12:53 AM - vo.x (Vit Ondruch)

trans (Thomas Sawyer) wrote:

Why keep the source code in the Ruby repository at all? Development occurs at the library's repository anyway, so why waste time with copying the files into Ruby repository for each release. Instead, just ship the .gem package with Ruby which can be installed via RubyGems (via rbininstall.rb) per the normal means of installing a gem.

Although I'd love to see Ruby shipping .gem files, the issue is that for example RubyGems needs Psych gem to work properly. So it is chicken/egg problem.

#43 - 11/29/2012 04:30 AM - sdaubert (Sylvain Daubert)

trans (Thomas Sawyer) wrote:

Why keep the source code in the Ruby repository at all? Development occurs at the library's repository anyway, so why waste time with copying the files into Ruby repository for each release. Instead, just ship the .gem package with Ruby which can be installed via RubyGems (via rbininstall.rb) per the normal means of installing a gem.

A full bundle may be easier installed when no internet connection is available (which often happens to me).

#44 - 11/29/2012 06:17 AM - trans (Thomas Sawyer)

[sdaubert \(Sylvain Daubert\)](#) That shouldn't be a problem in this case. All the necessary gems would be distributed with the Ruby source (it would sort of act like it's own miniature gem server, figuratively speaking).

#45 - 11/29/2012 06:23 AM - trans (Thomas Sawyer)

[vo.x \(Vit Ondruch\)](#) Psych would have to be an exception. And really, with the inclusion of libyaml, YAML is becoming more and more an integral aspect of Ruby.

#46 - 11/29/2012 07:22 AM - vo.x (Vit Ondruch)

[trans \(Thomas Sawyer\)](#) You know how it is with exceptions. They tend to become rules and justification for another exceptions. I believe that my proposal is the best solution, because it will be almost like shipping already preinstalled gems, which you would need to rebuild anyway. Packaged gems do not give you any additional benefit.

Anyway, I am not decision maker unfortunately. I just trying rectify current situation and prevent worsen solutions by being proactive :)

#47 - 11/29/2012 08:49 AM - trans (Thomas Sawyer)

[vo.x \(Vit Ondruch\)](#) Don't count your eggs before they hatch ;) I think the exceptions will be pretty well defined, basically anything RubyGems requires --isn't that going to be the case regardless of which way it's done? I thought packaged gems would be a benefit b/c one would not have to keep source files in sync with library repos, nor make particularly exceptional code for installing "fake" gems vs. real ones.

Understood. I'm just trying to flesh out this approach so whomever makes the decision has it for consideration.

#48 - 01/06/2013 06:58 AM - trans (Thomas Sawyer)

What is the status of this? I had a couple of thoughts/questions:

- I was wondering how standard lib like ostruct.rb fits into this. I did not notice it mentioned. Would it be a gem? Or are some standard libs staying "old-fashioned"?
- RubyGems dependencies seem to create a cache-22, as quoted "net/http, fileutils, optparse, psych or zlib are released as gems uninstalling them breaks RubyGems and you will not be able to reinstall them without reinstalling ruby." This makes these libs kind of quasi-core. Maybe an effort should be made to try to reduce RubyGems dependencies on these in so far as it is possible, and whatever is not possible should just be accepted as core?
- It still seems to me that maintenance of this would be immensely relieved if Ruby just bundled the set of standard gem packages, rather than redistribute copied source.

#49 - 01/31/2013 10:57 PM - vo.x (Vit Ondruch)

BTW there should be process in place which ensures, that the libraries are properly versioned, i.e. if something is changed, it cannot have the same version as already released library (see [#7761](#) and [#7762](#)).

#50 - 05/31/2014 03:21 AM - hsbt (Hiroshi SHIBATA)

- Assignee changed from nahi (Hiroshi Nakamura) to hsbt (Hiroshi SHIBATA)

#51 - 10/29/2014 03:42 PM - naruse (Yui NARUSE)

- Related to Feature #9612: Gemify OpenSSL added

#52 - 12/17/2014 01:54 PM - vo.x (Vit Ondruch)

- Related to Bug #10610: "make install" fails without zlib added

#53 - 04/02/2015 12:41 PM - vo.x (Vit Ondruch)

- Related to Feature #11025: Gemify Rake added

#54 - 04/11/2015 03:17 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #11057: Gemify JSON added

#55 - 03/09/2016 06:10 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #12160: Extract XMLRPC library to bundled gem added

#56 - 04/08/2016 02:54 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #8526: gemify tk added

#57 - 12/20/2016 03:44 AM - hsbt (Hiroshi SHIBATA)

After releasing Ruby 2.4.0 and Getting Matz approval, I will promote all of stdlibs to default gem.

I will ask this issue again at <https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20161221Japan>

#58 - 12/20/2016 04:50 AM - duerst (Martin Dürst)

Hiroshi SHIBATA wrote:

After releasing Ruby 2.4.0 and Getting Matz approval, I will promote all of stdlibs written by pure ruby to default gem.

Is this going to be part of 2.4 or part of 2.5?

Please be careful about lib/unicode_normalize.rb and lib/unicode_normalize/*. These are located in lib/, but they are part of String and built in.

#59 - 12/20/2016 06:34 AM - hsbt (Hiroshi SHIBATA)

Is this going to be part of 2.4 or part of 2.5?

part of 2.5.

Please be careful about lib/unicode_normalize.rb and lib/unicode_normalize/*.

I see. Thank you.

#60 - 12/20/2016 11:59 AM - zverok (Victor Shepelev)

What would be the status of ext/date and related things?

As far as I can understand, it is considered unmaintained/outdated currently, while providing not only Date class (useful), and DateTime (moderately useful, duplicating internal Time being incompatible with it), but also Date._parse, which is used by lib/time and entire Ruby world...

#61 - 12/26/2016 12:41 AM - hsbt (Hiroshi SHIBATA)

- Related to Misc #13072: Current state of date standard library added

#62 - 01/19/2017 06:02 AM - mrkn (Kenta Murata)

- Description updated

#63 - 01/19/2017 06:12 AM - matz (Yukihiro Matsumoto)

- Description updated

Go ahead and keep gemifying libraries discussing with their maintainer.

Matz.

#64 - 01/31/2017 08:46 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13173: Gemify webrick added

#65 - 02/01/2017 09:10 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13177: Gemify csv added

#66 - 02/02/2017 07:28 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13182: Gemify cmath added

#67 - 02/02/2017 08:36 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13183: Gemify date added

#68 - 02/08/2017 09:56 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13197: Gemify fileutils added

#69 - 02/08/2017 09:56 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13201: Gemify dbm added

#70 - 02/08/2017 09:57 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13199: Gemify strscan added

#71 - 02/08/2017 09:57 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13186: Gemify zlib added

#72 - 02/08/2017 09:57 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #11083: Gemify net-telnet added

#73 - 02/08/2017 09:58 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #8539: Unbundle ext/tk added

#74 - 02/08/2017 09:58 AM - hsbt (Hiroshi SHIBATA)

- Related to Bug #6123: Properly gemify BigDecimal added

#75 - 02/08/2017 09:59 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #2635: Unbundle rdoc added

#76 - 02/11/2017 08:59 AM - shevegen (Robert A. Heiler)

Looks as if this ticket may soon be resolved/finished after 5 years. :)

The only one on the todo list above is "Related to Ruby trunk - Misc [#13072](#): Current state of date standard library", or is there something else missing that has not been listed/updated yet? (5 years is a long time to keep track of)

#77 - 02/13/2017 06:35 PM - headius (Charles Nutter)

Please coordinate with us on the JRuby team. The released gems will need to be loadable in JRuby, since people will start depending on specific versions, upgrading them independently, etc.

MOST of stdlib in JRuby is identical to that in CRuby, but we do have a small set of patches. The diff can be seen by comparing the `jrubby-ruby-2_4` branch on our fork at <http://github.com/jruby/ruby> with the release version of Ruby 2.4, or similar branch for Ruby 2.3.

Of special concern are any stdlib that have C extensions. For those we can support, we have Java versions, and those will need to be released as gems as well.

We don't want another bigdecimal issue where there's a core gem that can't load on JRuby.

#78 - 02/14/2017 11:15 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13213: Gemify scanf added

#79 - 02/14/2017 11:26 AM - hsbt (Hiroshi SHIBATA)

[headius \(Charles Nutter\)](#)

Thank you for your information. I have never seen jruby patches in jruby/ruby repository.

<https://github.com/jruby/ruby/commit/b34aa3d97872a70ca91d8e2dd481628444f33d5e>

I think that the above patch is a good way to merge and maintain together after becoming a gem.

Of special concern are any stdlib that have C extensions. For those we can support, we have Java versions, and those will need to be released as gems as well.

It's good to move same situation as psych gem.

#80 - 02/24/2017 06:41 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13248: Gemify gdbm added

#81 - 02/28/2017 07:43 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #13261: Gemify sdbm added

#82 - 03/14/2017 08:06 PM - marcandre (Marc-Andre Lafortune)

I'd like to gemify matrix & ostruct, but it's not clear to me what needs to be done besides creating the gemspec and how future versions are meant to be released.

#83 - 03/14/2017 08:57 PM - headius (Charles Nutter)

hsbt (Hiroshi SHIBATA) wrote:

Thank you for your information. I have never seen jruby patches in jruby/ruby repository.

<https://github.com/jruby/ruby/commit/b34aa3d97872a70ca91d8e2dd481628444f33d5e>

I think that the above patch is a good way to merge and maintain together after becoming a gem.

Many of these changes could be merged back into MRI. I can start this process if you think it would be a good idea. We would like to reduce the size of this patch as much as possible, to ease maintenance.

It's good to move same situation as psych gem.

I will look over the other gemify issues and update with JRuby details.

#84 - 03/15/2017 06:44 AM - hsbt (Hiroshi SHIBATA)

I'd like to gemify matrix & ostruct

marcandre, Thank you for your attention.

I try to clarify to gemification process for you. I will create issue of matrix and ostruct gemification.

#85 - 05/15/2017 03:08 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #9456: Include bin/racc with ruby added

#86 - 07/27/2017 12:52 AM - hsbt (Hiroshi SHIBATA)

- Related to Misc #13771: Digest, Ruby OpenSSL, OpenSSL v1.1.0 added

#87 - 08/16/2017 02:38 AM - zenspider (Ryan Davis)

Hi hi... if you all want a meta-gem that you can version along with your releases and have it point to all the originally bundled gems, I am happy to transfer the ruby gem over to you. I was trying to get this ball rolling a long time ago and I camped it for just this reason...

That way if an install gets messed up it should be as simple as `gem install ruby -v 2.5.3`.

#88 - 08/16/2017 07:57 AM - hsbt (Hiroshi SHIBATA)

zenspider

Thanks for your offer. meta-gem idea is useful for update default gems.

Could you transfer ownership of ruby namespace to me?

#89 - 08/16/2017 09:22 AM - rbjl (Jan Lelis)

I like the idea that you could "update your standard library", for example, you are using Ruby 2.5.2, but want the standard library of 2.6.1. (as long as every gem in 2.6.1 is compatible with your old version of Ruby), which would be possible then.

It needs some thoughts though:

- More maintenance required: The *ruby* gem must be created and released together with Ruby. As long there is no automated process for importing upstream gem repos (or vice versa, for gems where development happens in core), there is always the danger that the gems on rubygems.org and the actually included **default gems** might get out of sync - which would cause a lot of confusion
- Should the *ruby* gem include **bundled gems** or not?

#90 - 09/24/2017 01:45 AM - akihikodaki (Akihiko Odaki)

Hi, I have some feedback implying a possibility of unexpected dependency on features packaged in Ruby installation when their gem versions are intended to be used.

I use Bundler v1.13.7 provided by CentOS SCL. The version always uses openssl packaged in Ruby installation even if there is a bundled one. That is caused by an dependency of Bundler on openssl. The dependency is *implicitly* made by securerandom. It is not documented in openssl documentation, and the documentation securerandom just says it may use openssl as random generator.

Though the dependency is removed by <https://github.com/bundler/bundler/commit/6167ec85e9ef4e9146fad305864b8359d4a9b021>, the commit message describes the change as performance improvement, showing the developers were not aware of the dependency on openssl and it was locked-in. I searched Bundler's issue tracker for related issues but had no luck.

This case reveals a fact that features packaged in Ruby installation can unexpectedly be used, and users may not be aware of that. It is problematic especially for such a security-related one. The issue points out security fixes as motivation, but it is hardly *secure* when it can be implicitly disabled.

Here I suggest two possible solutions for the problem:

- Document dependencies of standard libraries on gemified features
- Let RubyGems and Bundler warn if a gemified feature is already loaded

The later solution should actually be done in RubyGems and Bundler, but I decided to share it in this issue since it is the only reason such a warning should be implemented. I have not posted the feedback on the issue tracker of RubyGems nor one of Bundler, so they should be notified later if you think the problem should be handled by them.

#91 - 11/22/2017 10:51 PM - headius (Charles Nutter)

A few quick notes:

- Clarify for me please: are the new gemified stdlibs all going to use the default gem mechanism?
- Any "ruby" meta-gem will ideally need an equivalent for JRuby, so it doesn't try to install exts that don't exist for JRuby. It's very important that we don't introduce a bunch of new dependencies to the Ruby world that will be impossible for JRuby to support. We already have issues with folks depending on "openssl" which we do support but which does not have JRuby's pieces in the gem.
- I will begin trying to get out patches for stdlib libraries into each of the now-gemified libraries.

#92 - 11/23/2017 12:26 AM - MSP-Greg (Greg L)

Charles,

Clarify for me please: are the new gemified stdlibs all going to use the default gem mechanism?

Default gems have their code included in the ruby repo, and they cannot be removed from a user's installation. But, the user can install newer versions. Compared to an 'old style' stdlib, the main (only?) distinction is that a gemspec file exists, which is placed in the lib/ruby/gems/2.5.0/specifications/default folder, allowing RubyGems to install updated versions.

Bundled gems are not included in the repo, and are simply installed by the build scripts.

I believe a current list of trunk default & bundled gems can be found at the bottom of [appveyor-ruby](#).

Maybe this helps... Greg

#93 - 11/28/2017 05:35 AM - headius (Charles Nutter)

Greg, thanks! A few more questions.

Default gems have their code included in the ruby repo

I'm sure the default gem logic has evolved since I added it to RubyGems, but I'm curious about the gemspec that also lives alongside the library in lib/. This appears to be a new development. I have not followed RubyGems closely.

I may experiment with moving more things to gems on a 2.5 branch of JRuby so I can start to figure out what all has changed from previous releases.

#94 - 11/28/2017 08:24 AM - Hanmac (Hans Mackowiak)

MSP-Greg (Greg L) wrote:

I believe a current list of trunk default & bundled gems can be found at the bottom of [appveyor-ruby](#).

Maybe this helps... Greg

interesting list, i find it interesting that between this versions, some gems switched between "Default Gems" and "Bundled Gems"
2.3.5p376 => 2.4.2p198 => ruby 2.5.0dev

i think it has something to do with that you can remove bundled gems, but not default gems

#95 - 11/28/2017 01:57 PM - MSP-Greg (Greg L)

Hanmac (Hans Mackowiak) wrote:

i think it has something to do with that you can remove bundled gems, but not default gems

Yes, that's true when using standard ruby utilities.

Notice that I did say 'at the bottom of'. Both Travis and Appveyor have non-standard ruby builds, I assume based on what they feel their users want.

The trunk build on Appveyor is 'untouched'. Similar info at <https://travis-ci.org/MSP-Greg/travis-ruby>. Note that Travis does their trunk build.

#96 - 12/01/2017 05:01 AM - hsbt (Hiroshi SHIBATA)

Hi, [headius \(Charles Nutter\)](#).

- Clarify for me please: are the new gemified stdlibs all going to use the default gem mechanism?

I promoted following libraries to default gems at Ruby 2.5

```
cmath csv date dbm digest etc fcntl fiddle fileutils gdbm ipaddr scanf sdbm stringio strscan webrick zlib
```

- Any "ruby" meta-gem will ideally need an equivalent for JRuby, so it doesn't try to install exts that don't exist for JRuby. It's very important that we don't introduce a bunch of new dependencies to the Ruby world that will be impossible for JRuby to support. We already have issues with folks depending on "openssl" which we do support but which does not have JRuby's pieces in the gem.

I see. But I don't have "ruby" namespace yet. I remember that it needs to support JRuby platform.

- I will begin trying to get out patches for stdlib libraries into each of the now-gemified libraries.

Please see a repository list: <https://github.com/ruby/ruby/commit/9ffa42b995bc1ef9c2b763453f29d42a59755426>

I appreciate to merge JRuby code to our default gems. I hope to default gems helps your and JRuby users.

#97 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#98 - 07/10/2018 04:18 AM - marcandre (Marc-Andre Lafortune)

Thank you for putting matrix and ostruct as gems :-)

I'm not sure where to commit changes to the libraries now: do the commits need to be done in the gem's github repo *and* in MRI's repo? (Sorry, I couldn't find instructions!)

#99 - 07/14/2018 01:55 AM - hsbt (Hiroshi SHIBATA)

[marcandre \(Marc-Andre Lafortune\)](#)

do the commits need to be done in the gem's github repo and in MRI's repo?

You can commit it only github or MRI repo(SVN).

When you did commit code to github repo only, I merge them to MRI repo each week or month. (The case of MRI to GitHub is same instructions.)

I'm developing a merge tool between github and MRI repo and preparing an instruction guide about gemification until Ruby 2.6 release.

#100 - 07/14/2018 11:18 AM - Eregon (Benoit Daloze)

hsbt (Hiroshi SHIBATA) wrote:

You can commit it only github or MRI repo(SVN).

Good to know, so contributing to the gem's GitHub repo or directly in MRI are both supported?

When you did commit code to github repo only, I merge them to MRI repo each week or month. (The case of MRI to GitHub is same instructions.)

I'm developing a merge tool between github and MRI repo [...]

This sounds a bit similar to the synchronization with ruby/spec.

For ruby/spec I preserve all history from downstream repos (MRI, JRuby, TruffleRuby) when importing to ruby/spec, and in the other direction I update specs as a single commit in downstream repos.

Maybe <https://github.com/ruby/mspec/blob/master/tool/sync/sync-rubyspec.rb> can be useful as a base for your merge tool.

Feel free to take any code from there, it's all written by me :)

#101 - 09/18/2019 11:13 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #16170: Remove the unmaintained libraries from Ruby 2.7 added

#102 - 12/23/2019 12:33 AM - marcandre (Marc-Andre Lafortune)

[hsbt \(Hiroshi SHIBATA\)](#) is your tool to do this published somewhere?

In particular I noticed that some gemspec files are not in the same spot (e.g. lib/matrix/matrix.gemspec vs lib/ostruct.gemspec).

I'm moving one in the main repo, but it's not clear if the tool automatically does a search or if the correspondance is hardcoded.

I'm was actually wondering why they are in the main repository at all...

#103 - 12/23/2019 12:00 PM - hsbt (Hiroshi SHIBATA)

[marcandre \(Marc-Andre Lafortune\)](#)

You can pick the commits from ruby/* repository like this.

```
# git clone from the upstream repository and copy from the upstream repository with cp -rf. It's lost the git commits.
```

```
ruby tool/sync_default_gems.rb rubygems
```

```
# pick the single commit from the upstream repository.
```

```
ruby tool/sync_default_gems.rb rubygems 97e97686123e7338db628377d2f01516c8f9738f
```

```
# pick the commits range from the upstream repository
```

```
ruby tool/sync_default_gems.rb rubygems 97e97686123e7338db628377d2f01516c8f9738f..9e53702832
```

Sorry, your inconvenience experience. So, there is no enough document about default gems maintenance. I will prepare that in the next year.

#104 - 12/26/2019 10:00 PM - vo.x (Vit Ondruch)

marcandre (Marc-Andre Lafortune) wrote:

I'm was actually wondering why they are in the main repository at all...

The problem of .gemspec files is that they are scattered all across the repository. The .gemspec should live either in the root repository directory. That would allow to keep the upstream .gemspecs without modifications.

#105 - 12/27/2019 08:33 PM - marcandre (Marc-Andre Lafortune)

[hsbt \(Hiroshi SHIBATA\)](#) ah, thanks, I didn't know about ruby tool/sync_default_gems.rb. I doubled checked the code and the algorithm currently is in the subdirectory iff there is one, so what I did was correct and sufficient:

https://github.com/ruby/ruby/blob/c74d30e7950f18fd9fadcf404790e8fb0e891fd/tool/sync_default_gems.rb#L364-L368

vo.x (Vit Ondruch) wrote:

I'm was actually wondering why they are in the main repository at all...

That would allow to keep the upstream .gemspecs without modifications.

I'm sorry, I don't understand.

#106 - 12/27/2019 08:35 PM - marcandre (Marc-Andre Lafortune)

[hsbt \(Hiroshi SHIBATA\)](#) wrote (over 1 year ago):

[marcandre \(Marc-Andre Lafortune\)](#)

do the commits need to be done in the gem's github repo and in MRI's repo?

You can commit it only github or MRI repo(SVN).

When you did commit code to github repo only, I merge them to MRI repo each week or month.
(The case of MRI to GitHub is same instructions.)

There was an issue with prime library not beeing synced, I haven't checked others yet. It would be important to know if we can automatize this, and who's responsibility this is until we do...

<https://github.com/ruby/prime/issues/9>

#107 - 12/27/2019 09:09 PM - vo.x (Vit Ondruch)

marcandre (Marc-Andre Lafortune) wrote:

vo.x (Vit Ondruch) wrote:

I'm was actually wondering why they are in the main repository at all...

That would allow to keep the upstream .gemspecs without modifications.

I'm sorry, I don't understand.

You can compare the bundler.gemspec from Ruby and upstream:

<https://github.com/ruby/ruby/blob/master/lib/bundler/bundler.gemspec>
<https://github.com/bundler/bundler/blob/master/bundler.gemspec>

which are not the same. You can look at the other .gemspec files and compare them with the upstream versions.

You can also see the rinstall.rb logic handling all this, e.g.

<https://github.com/ruby/ruby/blame/master/tool/rinstall.rb#L662>
<https://github.com/ruby/ruby/blob/master/tool/rinstall.rb#L813>

which is full of various exceptions and heuristics. And there are other various issues due to the strange placement of the .gemspec files and their content:

<https://github.com/bundler/bundler/pull/6985>

IOW it would be easier, if Ruby shipped with the upstream .gemspec files on the level where they are upstream. In the gems upstream repositories, the .gemspec files are typically in root directory and the library files in lib/ subdirectory, which would perfectly match with the Ruby directory hierarchy.

BTW there is my PR trying to use more RubyGems functionality to install the default gems replacing the custom rbinstall code:

<https://github.com/ruby/ruby/pull/2545>

where I mention prime.gemspec and irb.gemspec which are unnecessarily modified.

#108 - 01/06/2020 06:35 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #15657: Make webrick to bundled gems added

#109 - 04/11/2020 10:21 PM - hsbt (Hiroshi SHIBATA)

- Related to Misc #16778: Should we stop vendoring default gems code? added

#110 - 05/01/2020 08:31 AM - marcandre (Marc-Andre Lafortune)

While tool/sync_default_gems can be useful to overwrite files, in some circumstances it won't help (e.g. different changes in both directories). I also wanted to move commits across repositories and not loose history. I found a solution that might benefit others, using the power of git.

My setup is same as for tool/sync_default_gems, i.e. the main ruby repo in ~/ruby/something and ~/ruby/matrix is the matrix library repository; change the paths below accordingly.

First time setup in the main repo:

```
git remote add lib_matrix ~/ruby/matrix/.git
git config remote.lib_matrix.tagopt --no-tags
```

Also do the same in the library repo if you need to bring commits the other way (i.e. from the main repo to the library)

This can seem quite strange because we are adding a "remote" that is a local git repository and that also has a completely unrelated history! But this makes it possible for git to have access to both histories at the same time.

To bring a commit from the library repository:

```
git fetch lib_matrix
git format-patch --stdout -1 <commit hash> | git am -3
```

This seems to automagically recognize the different paths from one repo to the other (e.g. lib/matrix/matrix.gemspec vs matrix.gemspec) and apply the patch correctly. git is awesome ♥️

If the commit in question changes more than just the library, you must limit the produced patch to just the desired files by specifying them. For example:

```
git format-patch --stdout -1 9f1fb0a17f lib/matrix* test/matrix | git am -3
```

HTH

Files

5481.pdf	78.2 KB	07/01/2012	nahi (Hiroshi Nakamura)
----------	---------	------------	-------------------------