Ruby trunk - Feature #5056

About 1.9 EOL

07/20/2011 12:03 AM - shyouhei (Shyouhei Urabe)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

=begin

At RubyKaigi, I was surprised to hear Matz saying "there will be no 1.9.4 because it becomes 2.0".

Question 1: are you kidding? or seriously speaking?

Question 2: do you have plan(s) for making 1.9 branch just like we have 1.8 branch now? or the whole 1.9 series just die when 2.0 development starts?

Question 3: who take care of the 2.0 branch? and who for 1.9 (if any)? Currently yugui is the mentor of 1.9 series. Does she shift to 2.0 mentor and new 1.9 person to appear, or she remains to 1.9 and new one for 2.0?

=end

History

#1 - 07/20/2011 09:53 AM - mame (Yusuke Endoh)

Hello,

2011/7/20 Shyouhei Urabe shyouhei@ruby-lang.org:

> At RubyKaigi, Â I was surprised to Â hear Matz saying "there Â will be no 1.9.4 because it becomes 2.0".
>
> Question 1: are you kidding? Â or seriously speaking?
>
> Question 2: Â do you have Â plan(s) for making Â 1.9 branch just Â like we have Â 1.8 branch Â now? Â or Â the Â whole 1.9 Â series just Â die when Â 2.0 development starts?
>
> Question 3: who take care of the 2.0 branch? and who for 1.9 (if any)? Currently yugui Â is the mentor of Â 1.9 series. Â Does she Â shift to 2.0 mentor and new 1.9 person to appear, or she remains to 1.9 and new one for 2.0?

I have no right to answer these questions, but if matz is planning to release 2.0 in near future (about three years?), I want to be 2.0 release manager!

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 07/20/2011 09:53 AM - sorah (Sorah Fukumori)

- Subject changed from About 1.9 EOL to About 1.9 EOL

- Assignee set to matz (Yukihiro Matsumoto)

#3 - 07/20/2011 10:50 AM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

I think the title of this topic is not "About 1.9 EOL" but "Road to 2.0".

First of all we should decide what is the 2.0.

If 2.0 is the next release of Ruby (it is called 1.9.4), it is only the naming issue.
(because I want to release next version of 1.9.3 in the year 2012 (or early 2013).)

If 2.0 is the ideal version of Ruby, matz must define what is the ideal version.
For example it includes

- Generational GC
- MVM
- Class box
- Refinements
- and so on

Whether creating ruby_1_9 branch or not, the life time of 1.9.[23] can be discussed after it is decided.

#4 - 07/20/2011 05:57 PM - shyouhei (Shyouhei Urabe)

> I think the title of this topic is not "About 1.9 EOL" but "Road to 2.0".

No, sorry. I know your "mood" but I'm honestly not interested in that area. Please make another thread for it.

What I'm worrying about here, is the futures of those 1.9.x branches.

#5 - 07/20/2011 11:04 PM - naruse (Yui NARUSE)

Shyouhei Urabe wrote:

> > I think the title of this topic is not "About 1.9 EOL" but "Road to 2.0".
>
> No, sorry. I know your "mood" but I'm honestly not interested in that area. Please make another thread for it.
>
> What I'm worrying about here, is the futures of those 1.9.x branches.

I believe the future of 1.9.x series depends whether 2.0 is one of 1.9 series or not.

#6 - 07/21/2011 01:39 AM - shyouhei (Shyouhei Urabe)

Sorry, I don't get it. "2.0 is one of 1.9 series" ? Please explain a bit.

But I really want this topic to be concise. Thank you.

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

#7 - 07/21/2011 09:59 AM - ko1 (Koichi Sasada)

Hi,

I understand NARUSE-san's comment (feature of 2.0 is matter) and
Urabe-san's comment (needs a lot of time to discuss 2.0 features).

My suggestion is:

- release 1.9.4, as an extension of 1.9.3 by 2012
- during making 1.9.4, discuss 2.0 features (and maintenance policies)
- after 1.9.4, split 1.9 branch and start 2.0 on trunk

I think 2.0 is good point to throw away some compatibility issues (e.g.
C APIs). We need more time to discuss.

Regards,
Koichi
--
// SASADA Koichi at atdot dot net

#8 - 07/21/2011 10:05 AM - naruse (Yui NARUSE)

Shyouhei Urabe wrote:

> Sorry, I don't get it. "2.0 is one of 1.9 series" ? Please explain a bit.
>
> But I really want this topic to be concise. Thank you.
>
> 風呂敷広げて議論が発散するのは勘弁してください。

There are two principles of Ruby 2.0; in short,

- 2.0 is much different from 1.9.3
- 2.0 is not different from 1.9.3 so much

On latter one, 2.0 is one of 1.9.x series and we don't need neither ruby_1_9 branch and this thread.

So first of all we should decide what is the 2.0.
Current my understanding is, Ruby 2.0 is the one we release in 2012.

#9 - 07/21/2011 10:20 AM - mame (Yusuke Endoh)

Why was 1.9 named "1.9", not "2.0" ? Because some feature was missing?
Where can I refer the discussion of the time? Or can anyone remember?
I guess it will be helpful to discuss this ticket.

--
Yusuke Endoh mame@tsg.ne.jp

#10 - 07/21/2011 10:29 AM - kosaki (Motohiro KOSAKI)

> > Sorry, I don't get it. "2.0 is one of 1.9 series" ? Please explain a bit.
> >
> > But I really want this topic to be concise. Thank you.
> >
> > 風呂敷広げて議論が発散するのは勘弁してください。
>
> There are two principles of Ruby 2.0; in short,
>
> - 2.0 is much different from 1.9.3
> - 2.0 is not different from 1.9.3 so much
>
> On latter one, 2.0 is one of 1.9.x series and we don't need neither ruby_1_9 branch and this >thread.
>
> So first of all we should decide what is the 2.0.
> Current my understanding is, Ruby 2.0 is the one we release in 2012.

OK, I've caught your point and I like this. I would suggest

- 1.9.4 will be released in early 2012. It has only small update. because development time is smaller than 1.9.[123].
- 2.0 will be released in 2013 Feb. it's good candidate because ruby was born at Feb 24 1993.
- 2.0 don't have any incompatibility
- no ruby_1_9 branch
- keep "release once per a year" rule
- 3.0 may have API change, but it's 2015 or later

Thought?

#11 - 07/21/2011 10:23 PM - Eregon (Benoit Daloze)

Hello,
On 21 July 2011 02:55, SASADA Koichi ko1@atdot.net wrote:

> I think 2.0 is good point to throw away some compatibility issues (e.g.
> C APIs). We need more time to discuss.

On 21 July 2011 03:05, Yui NARUSE naruse@airemix.jp wrote:

> There are two principles of Ruby 2.0; in short,
>
> - 2.0 is much different from 1.9.3
> - 2.0 is not different from 1.9.3 so much

On 21 July 2011 03:29, Motohiro KOSAKI kosaki.motohiro@gmail.com wrote:

> Â - 2.0 don't have any incompatibility
>
> Thought?

I believe 2.0 would be an occasion to do some clean up,
to throw away some compatibility issues when it makes sense to remove
unwanted features/API.

But at the same, I think 2.0 should not be so different from 1.9,
because we do not want another 1.8/1.9-like transition.

I realize I should probably not reply here, but the discussion is
somewhat hard to follow.
I like Motohiro's plan, except the "no incompatibility" rule.

#12 - 07/21/2011 11:23 PM - Anonymous

On Jul 21, 2011, at 8:22 AM, Benoit Daloze wrote:

> Hello,
> On 21 July 2011 02:55, SASADA Koichi ko1@atdot.net wrote:
>
> > I think 2.0 is good point to throw away some compatibility issues (e.g.
> > C APIs). We need more time to discuss.

This is a good opportunity to eliminate dangerous C API features like RHASH, RHASH_TBL, RSTRING_PTR, etc. Right now these features encourage dangerous access to internal runtime structures and cause a *lot* of issues for native threads.

cr

#13 - 07/21/2011 11:53 PM - now (Nikolai Weibull)

On Thu, Jul 21, 2011 at 16:22, Chuck Remes cremes.devlist@mac.com wrote:

> On Jul 21, 2011, at 8:22 AM, Benoit Daloze wrote:
>
> > On 21 July 2011 02:55, SASADA Koichi ko1@atdot.net wrote:
> >
> > > I think 2.0 is good point to throw away some compatibility issues (e.g.
> > > C APIs). We need more time to discuss.
>
> This is a good opportunity to eliminate dangerous C API features like RHASH, RHASH_TBL, RSTRING_PTR, etc. Right now these features encourage dangerous access to internal runtime structures and cause a *lot* of issues for native threads.

And a good opportunity to eliminate dangerous Ruby API features like
String#replace, String#sub!, and so on. Oh, to think a string
immutable, such a beauty it'd be.

#14 - 07/22/2011 12:01 AM - jonforums (Jon Forums)

I would like to see 2.0 clean up performance regressions on Windows introduced by 1.9. Cleaning up the Windows performance regressions is important for 2.0 and for removing a key roadblock for Windows users transitioning from 1.8 to MRI 1.9 or 2.0.

While work is ongoing to address require performance issues, I'm not aware of anything being done in the Windows IO area. This post is a summary of what I've found on file read performance between different MRI versions and JRuby. I'll include Rubinius once I get it to build on Win7.

I've created a rudimentary benchmarking/tracing/profiling helper tool at https://github.com/jonforums/measurements and the micro-benchmark results (disk based, not RAMdisk) are generated by two of the core workloads. Given the nature of micro-benchmarks, the helper tool really needs independent review to see if it's giving bogus results[1]

With those caveats, here are my current raw results https://gist.github.com/1097249 All MRI Rubies were built on Win7 32bit with the MinGW-based RubyInstaller recipes, and JRuby is their plain vanilla download.

Two interesting results:

1) For binary file reads (CRLF or LF), MRI 1.9.x outperforms both MRI 1.8.7 and JRuby 1.6.3 in 1.8.7 mode. Nice work!

2) For non-binary file reads (CRLF or LF), MRI 1.9.x is *dramatically* slower than both MRI 1.8.7 and JRuby 1.6.3 in 1.8.7. Below are the results. When it takes 1.9.2-p290 20.15s (real time) to do what MRI 1.8.7 does in 2.56s and JRuby 1.6.3 does in 3.62s, something is fundamentally wrong. Particularly if normal usage by libraries and user code opens files for reading in 'r' rather than 'rb' mode. The same problem persists in MRI 1.9.4dev.

microbenchmark for normal reading of file with CRLF endings

C:\projects\measurements-git>pik run rci bench core_rd_filelines_crlf

jruby 1.6.3 (ruby-1.8.7-p330) (2011-07-07 965162f) (Java HotSpot(TM) Client VM 1.6.0_26) [Windows 7-x86-java]
Rehearsal --
core_rd_filelines_crlf 3.853000 0.000000 3.853000 (3.826000)
-- total: 3.853000sec

```
                              user system total real
```

core_rd_filelines_crlf 3.629000 0.000000 3.629000 (3.628000)

ruby 1.8.7 (2011-06-30 patchlevel 352) [i386-mingw32]
Rehearsal --
core_rd_filelines_crlf 1.856000 0.156000 2.012000 (2.146123)
-- total: 2.012000sec

```
                              user system total real
```

core_rd_filelines_crlf 1.779000 0.281000 2.060000 (2.560147)

ruby 1.9.2p290 (2011-07-09 revision 32478) [i386-mingw32]
Rehearsal --
core_rd_filelines_crlf 19.781000 0.187000 19.968000 (20.216156)
-- total: 19.968000sec

```
                              user system total real
```

core_rd_filelines_crlf 19.672000 0.156000 19.828000 (20.153152)

ruby 1.9.4dev (2011-07-21 trunk 32590) [i386-mingw32]
Rehearsal --
core_rd_filelines_crlf 18.236000 0.094000 18.330000 (18.392052)
-- total: 18.330000sec

```
                              user system total real
```

core_rd_filelines_crlf 17.675000 0.078000 17.753000 (17.846020)

I'm publishing this info much earlier than I normally would because (a) it needs more eyes reviewing for validity, and (b) I don't want the issue to appear too late in the 2.0 development cycle to resolve. I'd be more than happy if it turned out that "there's no performance regression, you're just doing it wrong!" While I dislike eating steaming plates of crow, I hate that MRI Ruby runs fundamentally slower on my Windows boxes than on my Arch and Ubuntu Server boxes.

I'm going to continue investigating, but sadly I'm not able to give it as much time as it needs to resolve due to real, paying work. I'm hoping the issue is interesting and challenging enough so that (a) it can attract others, and (b) MRI project leadership views the issue as important for MRI's success and resources appropriately.

Thanks for your review and consideration,
Jon

[1] https://github.com/jonforums/measurements/blob/master/lib/inquisitor.rb#L57-103
https://github.com/jonforums/measurements/blob/master/workloads/core_rd_filelines_crlf.rb
https://github.com/jonforums/measurements/blob/master/workloads/core_rd_filelines_lf.rb
https://github.com/jonforums/measurements/blob/master/workloads/core_brd_filelines_crlf.rb
https://github.com/jonforums/measurements/blob/master/workloads/core_brd_filelines_lf.rb

#15 - 07/22/2011 12:53 AM - Eregon (Benoit Daloze)

On 21 July 2011 16:22, Chuck Remes cremes.devlist@mac.com wrote:
On 21 July 2011 16:33, Nikolai Weibull now@bitwi.se wrote:
On 21 July 2011 17:01, Jon Forums redmine@ruby-lang.org wrote:

SIGH

I absolutely did not want this thread to have any specific.
Please stop adding specifics when not asked, and read the whole thread
before answering.

On 20 July 2011 10:57, Shyouhei Urabe shyouhei@ruby-lang.org wrote:

> > I think the title of this topic is not "About 1.9 EOL" but "Road to 2.0".
>
> No, sorry. I know your "mood" but I'm honestly not interested in that area. Please make another thread for it.
>
> What I'm worrying about here, is the futures of those 1.9.x branches.

I'm really sorry, I guess I started unintentionally this.

There however a real need for 2.0 ideas/proposals, but I'm not sure
the ML is a good place, and certainly not this thread.

#16 - 07/22/2011 12:59 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:38287] Re: [Ruby 1.9 - Feature #5056] About 1.9 EOL"
on Thu, 21 Jul 2011 09:55:48 +0900, SASADA Koichi ko1@atdot.net writes:

|I understand NARUSE-san's comment (feature of 2.0 is matter) and
|Urabe-san's comment (needs a lot of time to discuss 2.0 features).
|
|My suggestion is:
|- release 1.9.4, as an extension of 1.9.3 by 2012
|- during making 1.9.4, discuss 2.0 features (and maintenance policies)
|- after 1.9.4, split 1.9 branch and start 2.0 on trunk
|
|I think 2.0 is good point to throw away some compatibility issues (e.g.
|C APIs). We need more time to discuss.

I disagree. Without making a branch, we have to wait 2.0 works until
we release 1.9.4 in the year 2012. Or do you mean by above "discuss",
to make a separate repository for 2.0 and experiment?

```
                          matz.
```

#17 - 07/22/2011 08:23 PM - naruse (Yui NARUSE)

(2011/07/22 0:54), Yukihiro Matsumoto wrote:

> Hi,
>
> In message "Re: [ruby-core:38287] Re: [Ruby 1.9 - Feature #5056] About 1.9 EOL"
> on Thu, 21 Jul 2011 09:55:48 +0900, SASADA Koichi ko1@atdot.net writes:
>
> |I understand NARUSE-san's comment (feature of 2.0 is matter) and
> |Urabe-san's comment (needs a lot of time to discuss 2.0 features).
> |
> |My suggestion is:
> |- release 1.9.4, as an extension of 1.9.3 by 2012
> |- during making 1.9.4, discuss 2.0 features (and maintenance policies)
> |- after 1.9.4, split 1.9 branch and start 2.0 on trunk
> |
> |I think 2.0 is good point to throw away some compatibility issues (e.g.
> |C APIs). We need more time to discuss.
>
> I disagree. Without making a branch, we have to wait 2.0 works until
> we release 1.9.4 in the year 2012. Or do you mean by above "discuss",
> to make a separate repository for 2.0 and experiment?

What we need is not your thought about ko1's plan.
We need your grand plan, not detail.

--
NARUSE, Yui naruse@airemix.jp

#18 - 07/23/2011 09:23 AM - wyhaines (Kirk Haines)

If 2.0 is just a lot compatible iteration of 1.9.3, and could just as easily
be called 1.9.4, then why not just call it 1.9.4? Using the 2.0 version jump
implies a substantial bump in the language, with changes in api's and
compatibility. It is the perfect place to clean up things that can not go
into 1.9.4 because the change is too big (such as C API changes, or adding

class boxing). So, to me, it would make sense to release 1.9.4 in early 2012, while developing a roadmap for what 2.0 should include. Then after 1.9.4 is released, we fork to start the 2.0 branch while someone continues to maintain 1.9.4 (and probably 1.9.3) for bug fixes. This is similar to what Koichi-san suggested, I believe.

Kirk Haines
On Jul 20, 2011 7:30 PM, "Motohiro KOSAKI" kosaki.motohiro@gmail.com wrote:

> Issue #5056 has been updated by Motohiro KOSAKI.
>
> > Sorry, I don't get it. "2.0 is one of 1.9 series" ? Please explain a bit.
> >
> > But I really want this topic to be concise. Thank you.
> >
> > $BIwO$I_9-2F5DO@$,H/;6$9kN$O4*J[$7F/$@$5$$!#(B
>
> There are two principles of Ruby 2.0; in short,
>
> - 2.0 is much different from 1.9.3
> - 2.0 is not different from 1.9.3 so much
>
> On latter one, 2.0 is one of 1.9.x series and we don't need neither ruby_1_9 branch and this >thread.
>
> So first of all we should decide what is the 2.0.
> Current my understanding is, Ruby 2.0 is the one we release in 2012.

OK, I've caught your point and I like this. I would suggest

- 1.9.4 will be released in early 2012. It has only small update. because development time is smaller than 1.9.[123].
- 2.0 will be released in 2013 Feb. it's good candidate because ruby was born at Feb 24 1993.
- 2.0 don't have any incompatibility
- no ruby_1_9 branch
- keep "release once per a year" rule
- 3.0 may have API change, but it's 2015 or later

Thought?

Feature #5056: About 1.9 EOL
http://redmine.ruby-lang.org/issues/5056

Author: Shyouhei Urabe
Status: Assigned
Priority: Normal
Assignee: Yukihiro Matsumoto
Category: Project
Target version: 2.0

=begin

At RubyKaigi, I was surprised to hear Matz saying "there will be no 1.9.4 because it becomes 2.0".

Question 1: are you kidding? or seriously speaking?

Question 2: do you have plan(s) for making 1.9 branch just like we have 1.8 branch now? or the whole 1.9 series just die when 2.0 development starts?

Question 3: who take care of the 2.0 branch? and who for 1.9 (if any)? Currently yugui is the mentor of 1.9 series. Does she shift to 2.0 mentor and new 1.9 person to appear, or she remains to 1.9 and new one for 2.0?

=end

--
http://redmine.ruby-lang.org

#19 - 08/10/2011 10:42 PM - sorah (Sorah Fukumori)

Hi,

I think we should decide about version changing,
so we should ask Matz's schedule in his mind.

Matz, do you have schedule in your mind?
If you have, please let us know :-)

Thanks,
Shota Fukumori

#20 - 08/10/2011 11:23 PM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:38900] [Ruby 1.9 - Feature #5056] About 1.9 EOL"
on Wed, 10 Aug 2011 22:42:59 +0900, Shota Fukumori sorah@tubusu.net writes:

|Matz, do you have schedule in your mind?
|If you have, please let us know :-)

My opinion is that we should make 1_9 branch after release of 1.9.3.
Then we will move forward 2.0 works on the trunk. 2.0 works includes

- keyword argument support for method definitions
- Module#mix
- Module#prepend
- and others (refinement, classbox, or method shelter?)

Currently I don't plan no big change to C API, but Ko1 might have
different opinion, especially regarding MVM.

```
                          matz.
```

#21 - 08/15/2011 11:32 AM - naruse (Yui NARUSE)

Yukihiro Matsumoto wrote:

> In message "Re: [ruby-core:38900] [Ruby 1.9 - Feature #5056] About 1.9 EOL"
> on Wed, 10 Aug 2011 22:42:59 +0900, Shota Fukumori sorah@tubusu.net writes:
> |Matz, do you have schedule in your mind?
> |If you have, please let us know :-)
>
> My opinion is that we should make 1_9 branch after release of 1.9.3.

What is 1_9 branch for. will we release 1.9.4?
If not, we don't need such branch; it only brings troublesome 2.0->1.9 merges.

> Then we will move forward 2.0 works on the trunk. 2.0 works includes
>
> - keyword argument support for method definitions
> - Module#mix
> - Module#prepend
> - and others (refinement, classbox, or method shelter?)

2.0 MUST those features? or MAY?
The release schedule depends on it.

#22 - 08/17/2011 06:59 PM - headius (Charles Nutter)

On Thu, Jul 21, 2011 at 10:01 AM, Jon Forums redmine@ruby-lang.org wrote:

> jruby 1.6.3 (ruby-1.8.7-p330) (2011-07-07 965162f) (Java HotSpot(TM) Client VM 1.6.0_26) [Windows 7-x86-java]
> Rehearsal --
> core_rd_filelines_crlf 3.853000 0.000000 3.853000 (3.826000)
> -- total: 3.853000sec
>
>   user system total real
> core_rd_filelines_crlf 3.629000 0.000000 3.629000 (3.628000)

If this is at all perf-intensive, you should be running JRuby with
--server, which will use the faster "server" compiler in the JVM. It's
anywhere from 25-100% faster (1.25x to 2x) faster than "client" mode.

- Charlie

#23 - 08/23/2011 06:53 AM - ko1 (Koichi Sasada)

Hi,

Sorry for my late response.

(2011/08/10 7:18), Yukihiro Matsumoto wrote:

> |Matz, do you have schedule in your mind?
> |If you have, please let us know :-)
>
> My opinion is that we should make 1_9 branch after release of 1.9.3.
> Then we will move forward 2.0 works on the trunk. 2.0 works includes
>
> - keyword argument support for method definitions
> - Module#mix
> - Module#prepend
> - and others (refinement, classbox, or method shelter?)
>
> Currently I don't plan no big change to C API, but Ko1 might have
> different opinion, especially regarding MVM.

I think I need to give up the API (and more about data structure)
changes. It should be done at 3.0 or later if there is no time/no
person to consider.

And I want to propose the followings:

- Release Ruby 2.0 with new features.
- Release Ruby 1.9.4 with performance changes and bug fixes.

Advantage:

- We can concentrate on implementing new features on 2.0 and also can concentrate on improving quality on 1.9.4.
- If the discussion of new features are not closing, we can release 1.9.4 (I think it is most important (*1)).

Disadvantage:

- It is ambiguous that which branch we should apply bug fixes.

*1: It's depends on [ruby-core:38955].

Release management of Ruby 2.0 by Endo-san is awesome.
If there is no other person can manage 1.9.4, I'll do it (if I can...).

Regards,
Koichi
--
// SASADA Koichi at atdot dot net

#24 - 08/23/2011 08:23 PM - lucas (Lucas Nussbaum)

On 23/08/11 at 06:50 +0900, SASADA Koichi wrote:

> (2011/08/10 7:18), Yukihiro Matsumoto wrote:
>
> > My opinion is that we should make 1_9 branch after release of 1.9.3.
> > Then we will move forward 2.0 works on the trunk. 2.0 works includes
> >
> > - keyword argument support for method definitions
> > - Module#mix
> > - Module#prepend
> > - and others (refinement, classbox, or method shelter?)
> >
> > Currently I don't plan no big change to C API, but Ko1 might have
> > different opinion, especially regarding MVM.

> I think I need to give up the API (and more about data structure) changes. It should be done at 3.0 or later if there is no time/no person to consider.
>
> And I want to propose the followings:
>
> - Release Ruby 2.0 with new features.
> - Release Ruby 1.9.4 with performance changes and bug fixes.
>
> Advantage:
>
> - We can concentrate on implementing new features on 2.0 and also can concentrate on improving quality on 1.9.4.
> - If the discussion of new features are not closing, we can release 1.9.4 (I think it is most important (*1)).
>
> Disadvantage:
>
> - It is ambiguous that which branch we should apply bug fixes.

Hi,

While I am not a Ruby developer (I only do "indirect" work by working on Debian packaging), I have been involved in a large number of Free Software projects over the years, and know a fair bit about release management.

I think that the current way of managing branches and releases of Ruby is not optimal.
There are too many (active) branches: the ruby_1_8, ruby_1_8_6, ruby_1_8_7, ruby_1_9_1, ruby_1_9_2, ruby_1_9_3 and trunk branches all received commits during the last year. That causes several problems:

- developer manpower is split between those branches.
- it is hard to keep track of bugfixes between branches. A severe bug affecting ruby_1_9_* is likely to require a fix in ruby_1_9_2, ruby_1_9_3, and trunk. Sometimes bugfixes don't get backported everywhere, resulting in releases with open bugs. The release cycles are too long, leading to:
- "time-to-market" for new features that is likely to be demotivating for developers
- long stabilization periods (+ unclear release schedules)

The current situation looks a lot like what was happening in the end of the Linux 2.4 era, where most development was happening in the 2.5 branch.

I think that you should inspire from the release management of Linux 2.6, and use one-branch-per-feature rather than one-branch-per-release. Then, you could have a single integration branch (that would most likely be trunk), and make releases from this branch, since features will have time to mature a bit inside feature branches.

Using shorter release schedules would also probably help. The addition of new features will be more incremental (less new features per release), which will also reduce the stabilization periods. Several important projects now use a 6-month release cycle. Maybe that could work for Ruby too?

Lucas

#25 - 08/23/2011 08:23 PM - naruse (Yui NARUSE)

(2011/08/23 20:09), Lucas Nussbaum wrote:

> On 23/08/11 at 06:50 +0900, SASADA Koichi wrote:
>
> > (2011/08/10 7:18), Yukihiro Matsumoto wrote:
> >
> > > My opinion is that we should make 1_9 branch after release of 1.9.3. Then we will move forward 2.0 works on the trunk. 2.0 works includes
> > >
> > > - keyword argument support for method definitions
> > > - Module#mix
> > > - Module#prepend
> > > - and others (refinement, classbox, or method shelter?)
> > >
> > > Currently I don't plan no big change to C API, but Ko1 might have different opinion, especially regarding MVM.

> I think I need to give up the API (and more about data structure) changes. It should be done at 3.0 or later if there is no time/no person to consider.
>
> And I want to propose the followings:
>
> - Release Ruby 2.0 with new features.
> - Release Ruby 1.9.4 with performance changes and bug fixes.
>
> Advantage:
>
> - We can concentrate on implementing new features on 2.0 and also can concentrate on improving quality on 1.9.4.
> - If the discussion of new features are not closing, we can release 1.9.4 (I think it is most important (*1)).
>
> Disadvantage:
>
> - It is ambiguous that which branch we should apply bug fixes.

> Hi,
>
> While I am not a Ruby developer (I only do "indirect" work by working on Debian packaging), I have been involved in a large number of Free Software projects over the years, and know a fair bit about release management.
>
> I think that the current way of managing branches and releases of Ruby is not optimal.
> There are too many (active) branches: the ruby_1_8, ruby_1_8_6, ruby_1_8_7, ruby_1_9_1, ruby_1_9_2, ruby_1_9_3 and trunk branches all received commits during the last year. That causes several problems:
>
> - developer manpower is split between those branches.
> - it is hard to keep track of bugfixes between branches. A severe bug affecting ruby_1_9_* is likely to require a fix in ruby_1_9_2, ruby_1_9_3, and trunk. Sometimes bugfixes don't get backported everywhere, resulting in releases with open bugs. The release cycles are too long, leading to:
> - "time-to-market" for new features that is likely to be demotivating for developers
> - long stabilization periods (+ unclear release schedules)
>
> The current situation looks a lot like what was happening in the end of the Linux 2.4 era, where most development was happening in the 2.5 branch.
>
> I think that you should inspire from the release management of Linux 2.6, and use one-branch-per-feature rather than one-branch-per-release. Then, you could have a single integration branch (that would most likely be trunk), and make releases from this branch, since features will have time to mature a bit inside feature branches.
>
> Using shorter release schedules would also probably help. The addition of new features will be more incremental (less new features per release), which will also reduce the stabilization periods. Several important projects now use a 6-month release cycle. Maybe that could work for Ruby too?

You don't want patch release?

--
NARUSE, Yui naruse@airemix.jp

#26 - 08/23/2011 08:53 PM - shyouhei (Shyouhei Urabe)

Hello Lucas.

(08/23/2011 08:09 PM), Lucas Nussbaum wrote:

> I think that the current way of managing branches and releases of Ruby is not optimal.

Indeed. But I'm not sure if Linux-style release management works in this project. Ruby is Ruby, not Linux. Almost no programmers (except Matz) had been paid to run this project until recently. I doubt a 6-month release

cycle could hardly work for a hobby project like this.

#27 - 08/23/2011 09:23 PM - lucas (Lucas Nussbaum)

On 23/08/11 at 20:20 +0900, NARUSE, Yui wrote:

> (2011/08/23 20:09), Lucas Nussbaum wrote:
>
> > On 23/08/11 at 06:50 +0900, SASADA Koichi wrote:
> >
> > > (2011/08/10 7:18), Yukihiro Matsumoto wrote:
> > >
> > > > My opinion is that we should make 1_9 branch after release of 1.9.3.
> > > > Then we will move forward 2.0 works on the trunk. 2.0 works includes
> > > >
> > > > - keyword argument support for method definitions
> > > > - Module#mix
> > > > - Module#prepend
> > > > - and others (refinement, classbox, or method shelter?)
> > > >
> > > > Currently I don't plan no big change to C API, but Ko1 might have
> > > > different opinion, especially regarding MVM.
> > >
> > > I think I need to give up the API (and more about data structure)
> > > changes. It should be done at 3.0 or later if there is no time/no
> > > person to consider.
> > >
> > > And I want to propose the followings:
> > >
> > > - Release Ruby 2.0 with new features.
> > > - Release Ruby 1.9.4 with performance changes and bug fixes.
> > >
> > > Advantage:
> > >
> > > - We can concentrate on implementing new features on 2.0 and also can concentrate on improving quality on 1.9.4.
> > > - If the discussion of new features are not closing, we can release 1.9.4 (I think it is most important (*1)).
> > >
> > > Disadvantage:
> > >
> > > - It is ambiguous that which branch we should apply bug fixes.
> >
> > Hi,
> >
> > While I am not a Ruby developer (I only do "indirect" work by working on
> > Debian packaging), I have been involved in a large number of Free
> > Software projects over the years, and know a fair bit about release
> > management.
> >
> > I think that the current way of managing branches and releases of Ruby
> > is not optimal.
> > There are too many (active) branches: the ruby_1_8, ruby_1_8_6,
> > ruby_1_8_7, ruby_1_9_1, ruby_1_9_2, ruby_1_9_3 and trunk branches all
> > received commits during the last year. That causes several problems:
> >
> > - developer manpower is split between those branches.
> > - it is hard to keep track of bugfixes between branches. A severe bug affecting ruby_1_9_* is likely to require a fix in ruby_1_9_2, ruby_1_9_3, and trunk. Sometimes bugfixes don't get backported everywhere, resulting in releases with open bugs. The release cycles are too long, leading to:
> > - "time-to-market" for new features that is likely to be demotivating for developers
> > - long stabilization periods (+ unclear release schedules)
> >
> > The current situation looks a lot like what was happening in the end of
> > the Linux 2.4 era, where most development was happening in the 2.5
> > branch.
> >
> > I think that you should inspire from the release management of Linux
> > 2.6, and use one-branch-per-feature rather than one-branch-per-release.
> > Then, you could have a single integration branch (that would most likely
> > be trunk), and make releases from this branch, since features will have
> > time to mature a bit inside feature branches.
> >
> > Using shorter release schedules would also probably help. The addition
> > of new features will be more incremental (less new features per

> release), which will also reduce the stabilization periods. Several important projects now use a 6-month release cycle. Maybe that could work for Ruby too?

You don't want patch release?

It depends on your definition of 'patch releases'.

'patch releases' could be 'bugfixes-only' releases, such as the Linux 2.6.32.x releases, the GNOME stable releases, the Debian 'point releases', etc.

Currently, Ruby's definition of patch releases is a bit unclear to me: it contains bugfixes, but also new features, and sometimes regressions. You can't say for sure that 1.9.2p290 is better in all aspects than 1.9.2p180, because so many things have changed that it's unlikely that no regressions have been introduced.

So, in 'my' ideal world, we could get something like:

- every 6 months, a new release of Ruby, with some bugfixes, some new features, some regressions ;). This release is branched/tagged from the main development branch of Ruby (this ensures that no code stays in trunk for too long, without users).
- maybe, patch releases, which are for users who require a "rock-solid", absolutely stable Ruby. Those releases would only contain bugfixes.

You wouldn't need to do patch releases for every Ruby releases. You could adopt a "long term support" model, where some selected Ruby releases will be maintained for a long time.

Lucas

#28 - 08/23/2011 09:23 PM - lucas (Lucas Nussbaum)

On 23/08/11 at 20:38 +0900, Urabe Shyouhei wrote:

> Hello Lucas.
>
> (08/23/2011 08:09 PM), Lucas Nussbaum wrote:
>
> > I think that the current way of managing branches and releases of Ruby is not optimal.
>
> Indeed. But I'm not sure if Linux-style release management works in this project. Ruby is Ruby, not Linux. Almost no programmers (except Matz) had been paid to run this project until recently. I doubt a 6-month release cycle could hardly work for a hobby project like this.

Interesting. Why do you think so?

I don't really see a link between shorter release cycles and being able to work full time on a project. It's true that it is easier to meet deadlines when you work full-time on a project, but on the other hand, shorter release cycles releave some pressure from developers, because, if a feature can't make it into release 'n', it can still make it into release 'n+1' which will be released in 6 months (and not in two years).

Lucas

#29 - 08/23/2011 09:53 PM - shyouhei (Shyouhei Urabe)

(08/23/2011 09:20 PM), Lucas Nussbaum wrote:

> On 23/08/11 at 20:38 +0900, Urabe Shyouhei wrote:
>
> > Hello Lucas.
> >
> > (08/23/2011 08:09 PM), Lucas Nussbaum wrote:
> >
> > > I think that the current way of managing branches and releases of Ruby is not optimal.

> > Indeed. But I'm not sure if Linux-style release management works in this project. Ruby is Ruby, not Linux. Almost no programmers (except Matz) had been paid to run this project until recently. I doubt a 6-month release cycle could hardly work for a hobby project like this.
>
> Interesting. Why do you think so?

Because we once tried this: in the age of 1.8.2 through 1.8.5 (-p0). We observed that 6 months are a bit too short for a new toy. Relatively few people were going to look at forthcoming releases, which led many last-minute push to them, and thus, result in poor quality.

A hobby is a hobby because no one forces you to meet a deadline.

> I don't really see a link between shorter release cycles and being able to work full time on a project. It's true that it is easier to meet deadlines when you work full-time on a project, but on the other hand, shorter release cycles releave some pressure from developers, because, if a feature can't make it into release 'n', it can still make it into release 'n+1' which will be released in 6 months (and not in two years).

Full-timer or not is not that important. The point is we need something different from employee management. 6 months are too long for a bug to be fixed, while a bit too short for a hobbyist to look at.

It might be good for a feature like you say, but you know, man shall not live by feature alone.

#30 - 08/24/2011 06:23 PM - naruse (Yui NARUSE)

2011/8/23 Michal Suchanek hramrach@centrum.cz:

> On 23 August 2011 13:20, NARUSE, Yui naruse@airemix.jp wrote:
>
> > (2011/08/23 20:09), Lucas Nussbaum wrote:
> >
> > > On 23/08/11 at 06:50 +0900, SASADA Koichi wrote:
> > >
> > > > (2011/08/10 7:18), Yukihiro Matsumoto wrote:
> > > >
> > > > I think that you should inspire from the release management of Linux 2.6, and use one-branch-per-feature rather than one-branch-per-release. Then, you could have a single integration branch (that would most likely be trunk), and make releases from this branch, since features will have time to mature a bit inside feature branches.
> > > >
> > > > Using shorter release schedules would also probably help. The addition of new features will be more incremental (less new features per release), which will also reduce the stabilization periods. Several important projects now use a 6-month release cycle. Maybe that could work for Ruby too?
> > >
> > > You don't want patch release?
> >
> > If you use Linux for comparison you can see that it has patch releases.
> >
> > They have 2.6.N released from trunk and backport fixes to 2.6.(N-1).n.

Hmm, thanks for explanation.

As far as I understand, current Ruby's development model is the same one except the fast release cycle.

--
NARUSE, Yui Â naruse@airemix.jp

#31 - 08/24/2011 11:23 PM - yugui (Yuki Sonoda)

On Tue, Aug 23, 2011 at 9:20 PM, Lucas Nussbaum
lucas@lucas-nussbaum.net wrote:

> Currently, Ruby's definition of patch releases is a bit unclear to me:
> it contains bugfixes, but also new features, and sometimes
> regressions.

I'm trying to reject new features for patch releases. Could you give me an example of my mistake?

Also I'm trying to keep patch releases without regression. I know I did some mistakes in this area, but also I don't think your proposal prevents these kinds of mistakes.

--
Yuki Sonoda (Yugui)
yugui@yugui.jp
http://yugui.jp

#32 - 08/24/2011 11:29 PM - yugui (Yuki Sonoda)

On Thu, Jul 21, 2011 at 10:29 AM, Motohiro KOSAKI
kosaki.motohiro@gmail.com wrote:

> Â - 1.9.4 will be released in early 2012. It has only small update.
> Â because development time is smaller than 1.9.[123].
> Â - 2.0 will be released in 2013 Feb. it's good candidate because ruby was born at Feb 24 1993.
> Â - 2.0 don't have any incompatibility
> Â - no ruby_1_9 branch
> Â - keep "release once per a year" rule
> Â - 3.0 may have API change, but it's 2015 or later

Basically I agree with Motohiro, except:

> Â - no ruby_1_9 branch

Ruby 1.9 will be good enough with Ruby 1.9.3. Ruby 1.9.2 resolved some contradictive/confusing language designs in Ruby 1.9.1. Ruby 1.9.3 improved the implementation. So next, what should we do to make Ruby better?

- Deprecation of unwanted APIs/features
- Large enhancements, like keyword arguments, refinements or classbox.

Ruby with these changes should be called Ruby 2.0. Matz is right. But also these features will take some time. It cannot be released within 2012. So 2013 Feb is a good candidate.

On Fri, Jul 22, 2011 at 12:54 AM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

> I disagree. Â Without making a branch, we have to wait 2.0 works until
> we release 1.9.4 in the year 2012.

Yes. We should have a branch.

> Â - no ruby_1_9 branch
> But it can be ruby_1_9_4. Motohiro is also right. :)

--
Yuki Sonoda (Yugui)
yugui@yugui.jp
http://yugui.jp

#33 - 08/24/2011 11:29 PM - lucas (Lucas Nussbaum)

On 24/08/11 at 23:03 +0900, Yugui wrote:

> On Tue, Aug 23, 2011 at 9:20 PM, Lucas Nussbaum
> lucas@lucas-nussbaum.net wrote:

> Currently, Ruby's definition of patch releases is a bit unclear to me:
> it contains bugfixes, but also new features, and sometimes
> regressions.

I'm trying to reject new features for patch releases. Could you give me an example of my mistake?

We migrated from svn to git for Debian packaging and did not keep the history, so it's a bit hard to find exact references.

Looking at the recent history, 1.9.2 has been good in that regard, with the exception of the fix for CVE-2011-0188 that is missing in .290, which we had to backport from another branch (fix is r30993).
But that's not a good example, since it illustrates "hard to backport every relevant bugfix", not "patch releases containing new features".

However, in the 1.8.7 branch, several patch releases were containing user-visible changes that were not bugfixes, leading to incompatibilities (I think it was in .72, but I'm not sure anymore). If the policy is to not do that anymore, I'm very happy. :)

> Also I'm trying to keep patch releases without regression. I know I
> did some mistakes in this area, but also I don't think your proposal
> prevents these kinds of mistakes.

I don't know if it's the case in Ruby, but there are projects where there is some pressure from developers on release managers to push features to users. A shorter release cycle helps in that regard by making developers who want to push features to users happier.

Lucas

#34 - 08/25/2011 04:23 AM - lucas (Lucas Nussbaum)

On 24/08/11 at 23:27 +0900, Yugui wrote:

> On Thu, Jul 21, 2011 at 10:29 AM, Motohiro KOSAKI
> kosaki.motohiro@gmail.com wrote:
>
> > Â - 1.9.4 will be released in early 2012. It has only small update.
> > Â because development time is smaller than 1.9.[123].
> > Â - 2.0 will be released in 2013 Feb. it's good candidate because ruby was born at Feb 24 1993.
> > Â - 2.0 don't have any incompatibility
> > Â - no ruby_1_9 branch
> > Â - keep "release once per a year" rule
> > Â - 3.0 may have API change, but it's 2015 or later
>
> Basically I agree with Motohiro, except:
>
> > Â - no ruby_1_9 branch
>
> Ruby 1.9 will be good enough with Ruby 1.9.3. Ruby 1.9.2 resolved
> some contradictive/confusing language designs in Ruby 1.9.1. Ruby
> 1.9.3 improved the implementation. So next, what should we do to make
> Ruby better?
>
> - Deprecation of unwanted APIs/features
> - Large enhancements, like keyword arguments, refinements or classbox.
>
> Ruby with these changes should be called Ruby 2.0. Matz is right.
> But also these features will take some time. It cannot be released
> within 2012. So 2013 Feb is a good candidate.
>
> On Fri, Jul 22, 2011 at 12:54 AM, Yukihiro Matsumoto matz@ruby-lang.org wrote:
>
> > I disagree. Â Without making a branch, we have to wait 2.0 works until
> > we release 1.9.4 in the year 2012.
>
> Yes. We should have a branch.

> Â - no ruby_1_9 branch
> But it can be ruby_1_9_4. Motohiro is also right. :)

Hi,

Just to make sure I got it correctly:

After early 2012 and the release of 1.9.4, we would get:

- trunk => development branch to prepare 2.0
- ruby_1_9_3 branch => 'stable' branch, bugfix only, to prepare 1.9.3 patch releases
- ruby_1_9_4 branch => 'stable' branch, bugfix only, to prepare 1.9.4 patch releases

That's all? That sounds excellent to me.

If you can get 1.9.4 released in January 2012, then it can be included
in Ubuntu 12.04 (which is a long term support release). It might also
be possible if released in february. But March would be too late.

The next Debian stable release is supposed to freeze in June 2012. So it
will easily get 1.9.4.

- Lucas

#35 - 08/25/2011 08:53 PM - kosaki (Motohiro KOSAKI)

> I'm trying to reject new features for patch releases. Â Could you give
> me an example of my mistake?

We migrated from svn to git for Debian packaging and did not keep the
history, so it's a bit hard to find exact references.

Looking at the recent history, 1.9.2 has been good in that regard, with
the exception of the fix for CVE-2011-0188 that is missing in .290,
which we had to backport from another branch (fix is r30993).
But that's not a good example, since it illustrates "hard to backport
every relevant bugfix", not "patch releases containing new features".

However, in the 1.8.7 branch, several patch releases were containing
user-visible changes that were not bugfixes, leading to
incompatibilities (I think it was in .72, but I'm not sure anymore). If
the policy is to not do that anymore, I'm very happy. :)

Hi

I who linux kernel developer would like to explain this. In fact, Linux
has several stable branch maintainer and they have slightly different
maintenance policy. Similar, Ruby 1.8.7 and 1.9.2 have slightly different
policy. If you want and will become a branch maintainer of our community,
you may be able to have more different policy. That's authority and
responsibility
of a maintainer.

Anyway 1.8.7 is a really special exception, it's a final release of 1.8.x.
I don't think it's good example for discussing maintenance policy.

#36 - 08/25/2011 09:23 PM - kosaki (Motohiro KOSAKI)

> Just to make sure I got it correctly:
>
> After early 2012 and the release of 1.9.4, we would get:
>
> - trunk => development branch to prepare 2.0
> - ruby_1_9_3 branch => 'stable' branch, bugfix only, to prepare 1.9.3 patch releases
> - ruby_1_9_4 branch => 'stable' branch, bugfix only, to prepare 1.9.4 patch releases
>
> That's all? That sounds excellent to me.

> If you can get 1.9.4 released in January 2012, then it can be included in Ubuntu 12.04 (which is a long term support release). It might also be possible if released in february. But March would be too late.
>
> The next Debian stable release is supposed to freeze in June 2012. So it will easily get 1.9.4.

Hmm....

I don't think Jan 2012 is a good release target. It's too short development time.
And I hope 1.9.4 has ~2 month feature freezing length for keeping quality as past releases. So, I think March or April is best target. Jun is acceptable on boundary. Probably July is too late, it might make bad side effect to 2.0 release schedule.

#37 - 10/18/2011 09:16 AM - naruse (Yui NARUSE)

- Project changed from Ruby trunk to CommonRuby

- Category deleted (Project)

- Target version deleted (Next Major)

#38 - 10/23/2011 05:21 PM - naruse (Yui NARUSE)

- Project changed from CommonRuby to Ruby trunk

#39 - 11/24/2012 01:06 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

Is there any reason to keep this ticket open?
I'm closing. Please open separate ticket for each concrete issue (if any).

--
Yusuke Endoh mame@tsg.ne.jp