Ruby master - Feature #16345

Don't emit deprecation warnings by default.

11/12/2019 07:07 AM - akr (Akira Tanaka)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

We propose that Ruby doesn't emit deprecation warnings by default.

Deprecation warnings are only useful during development for updating Ruby version.
They are not useful during development with current Ruby.
It is especially frustrating when deprecated warnings are generated in gems.
Also, deprecation warnings are totally useless in production environment.

So, we want to emit deprecation warnings only in useful situations.

We propose a command line argument -W:deprecated (or --warning=deprecated)
and the following methods to enable/disable deprecation warnings.

```
Warning.disable(:deprecated)
Warning.enable(:deprecated)
Warning.enabled?(:deprecated)
```

Currently we don't propose a method to generate a deprecation warning
because currently our main intent is to disable deprecation warnings for
keyword arguments, and the warnings are generated in C level.

Background:

We talked about keyword arguments during a developer meeting (2019-11-12).
https://bugs.ruby-lang.org/issues/16333
We expect many deprecation warnings to be generated in Ruby 2.7.
They are not useful except for development for Ruby transition, and
they may block transition to Ruby 2.7.

So, we have consensus to disable deprecation warnings by default.
Our design is intentionally minimum because we need this feature for Ruby 2.7.

We chose Warning.disable(:deprecated) instead of
re-defining Warning.warn in order to avoid string object generation.

Of course, we expect to extend this feature:
Ruby-level deprecation warning generation,
warnings other than deprecation,
file-based restriction of warning generation, etc.
But this issue doesn't contain them.

Related issues:

Related to Ruby master - Feature #16289: Reduce duplicated warnings for the c...	**Closed**
Related to Ruby master - Feature #16018: Add a way to deprecate methods	**Open**
Related to Ruby master - Feature #17000: 2.7.2 turns off deprecation warnings...	**Closed**

Associated revisions

Revision a84ad243 - 12/20/2019 02:05 PM - nobu (Nobuyoshi Nakada)

Added -W: command line option

To manage Warning[category] flags. Only -W:deprecated and
-W:no-deprecated are available now. [Feature #16345]

Revision d76c8cfe - 12/22/2019 06:18 AM - nobu (Nobuyoshi Nakada)

RDoc of Warning.[] and .[]= [Feature #16345] [ci skip]

Revision 1ed87dd3 - 12/23/2019 02:08 AM - nobu (Nobuyoshi Nakada)

Add NEWS about Warning.[] [Feature #16345] [ci skip]

Revision 996af2ce - 09/25/2020 12:50 AM - nobu (Nobuyoshi Nakada)

Disable deprecation warning by the default [Feature #16345]

And -w option turns it on.

Revision df3f52a6 - 09/29/2020 01:43 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 996af2ce086249e904b2ce95ab2fcd1de7d757be: [Backport #16345] [Backport #17000]

```
    Disable deprecation warning by the default [Feature #16345]

    And `-w` option turns it on.
```

History

#1 - 11/12/2019 08:16 AM - duerst (Martin Dürst)

I like this feature because it would make it easier to get a grip on the various features in Ruby that are scheduled for deprecation. Currently, we don't have an overall picture of what is deprecated and when and how we plan to remove it. Ideally, deprecation warnings should come with additional information, i.e. version where the feature will be removed, and/or issue on this tracker where the details are.

I'm not sure making Warning.disable(:deprecated) the default is the best thing to do, because it won't help people who use Ruby just for simple scripting (without an explicit deploy process). It may be better for deployments to explicitly set Warning.enable(:deprecated).

#2 - 11/12/2019 09:21 AM - shevegen (Robert A. Heiler)

I have no huge, strong preference either way because I feel you can find use cases and advantages with either way. This taps into other suggestions about ruby users being able to control the verbosity of ruby warnings in general, e. g. to adjust them to their personal use cases. I myself run with -w all the time (I just got so used to it) but I understand other ruby users not running ruby code with -w, or wanting more fine tuned adjustment.

-W:deprecated or something like that seems fine; perhaps also an additional toggle-state via environment variables, like RUBY_VERBOSITY or some such (environment variables are not great, though, because it also requires of ruby users to know about them, and the more there are, the harder this becomes).

I think the main issue is that there are people who like the warning, including being warned about it by default. Martin gave a good explanation and I also agree with what he wrote. So I think this is simply a case of different valid use cases that are somewhat orthogonal to each other - one "side" can reason in favour, the other against it. :) The best may be to decide on a default that covers what seems important, and to then allow ruby users to set to what they want specifically.

By the way, on a side note - I think for ruby 3.0 it may be best to not change the default in this regard, and to change it past ruby 3.0. While warnings are not necessarily restricted to matz' goal of having the transition to ruby 3.0 very simple, it may still annoy some ruby users when they change to ruby 3.0 and suddenly have more verbose warning-chatter. (For me it would not matter, I can change my code either way.)

> Ideally, deprecation warnings should come with additional information, i.e. version where the feature will be removed, and/or issue on this tracker where the details are.

Agreed. For me this information would be useful since I can prepare for change that way. When frozen string: true/false was enabled and added to ruby initially, I did not have a lot of code that would work with frozen strings. Today almost all of my ruby code works with frozen strings set to true, and the remaining code that does not is mostly old legacy code that I can quickly update now. So transitioning now would be simple for me - but back when it was added, it would have been quite painful. It was simpler to spread the workload over several weeks; this helped avoidd frustration too. That is why deprecation notices should ideally come way in advance.

> I'm not sure making Warning.disable(:deprecated) the default is the best thing to do, because it won't help people who use Ruby just for simple scripting (without an explicit deploy process). It may be better for deployments to explicitly set Warning.enable(:deprecated).

I think you can find arguments in favour or in disfavour either way, depending on the use case and background of the ruby user at hand. It also depends on the warning itself. As said I use -w all the time, but some warnings are more useful than others. Some are a bit strange - first time I saw the incorrect indentation warning I wondered whether ruby was now like python. ;) It usually happens when I do a typo in my code, but I also wonder why the warning comes in general, and more importantly that we have no way to sort of "toggle" which warnings we want; we could use a model a bit similar to rubocop, although I should also add that configuring rubocop is not necessarily super-easy, so good defaults should be used for ruby whenever possible. This is mostly for a "be able to customize the warnings to your use case and style", which feels related to rubocop IMO.

I think the ideal would be to have some kind of war to toggle ruby's state and behaviour in regards to warnings on a per-ruby system-wide state, without depending on code directly embedded into the .rb file (although I am also in favour of being able to toggle the state here too). IMO allowing as much flexibility as possible here.

But I would suggest to change the default here after 3.0; it's not that long into the future anyway since 3.0 will come out next year.

#3 - 11/12/2019 09:22 AM - shevegen (Robert A. Heiler)

By the way on the API itself suggested by Akira, I think it seems sensible.

These ones I meant:

```
Warning.disable(:deprecated)
Warning.enable(:deprecated)
Warning.enabled?(:deprecated)
```

Although it could be another API perhaps, imo the use case is a fine one - to be able to customize ruby in this regard.

#4 - 11/12/2019 10:01 AM - sawa (Tsuyoshi Sawada)

- Description updated

#5 - 11/12/2019 02:52 PM - Dan0042 (Daniel DeLorme)

Personally I'd like to have this feature as an attribute accessor of Warning. So it's easier to set the value based on configuration.

```
Warning[:deprecated] = true
Warning[:deprecated] = ENV.include?('WARN_DEPRECATED')
```

#6 - 11/12/2019 11:08 PM - akr (Akira Tanaka)

Dan0042 (Daniel DeLorme) wrote:

> Personally I'd like to have this feature as an attribute accessor of Warning. So it's easier to set the value based on configuration.
>
> ```
> Warning[:deprecated] = true
> Warning[:deprecated] = ENV.include?('WARN_DEPRECATED')
> ```

We discussed about environment variable based configuration.
It is possible without application code as: RUBYOPT=-W:deprecated.

#7 - 11/12/2019 11:55 PM - Eregon (Benoit Daloze)

I think the ability to enable & disable deprecation is a great idea.

But I think disabling by default is almost equivalent to give up on anyone ever caring about these deprecation warnings, because most developers won't even see them.

I think we need to show deprecation warnings by default, or at least show one line of warning that deprecation occurred, but are hidden by default like:

```
warning: deprecated call occurred, use -W:deprecated to see details or -W:no-deprecated to silence this warnin
g
```

Also, I think -w should show deprecation warnings by default.

I'm not sure an API on Warning is useful, I think for most use cases using command line flags like -W:deprecated/-W:no-deprecated would be better (e.g., it's likely that some warnings are missed when doing Warning.enable(:deprecated) at runtime).

#8 - 11/13/2019 12:14 AM - akr (Akira Tanaka)

Eregon (Benoit Daloze) wrote:

> I'm not sure an API on Warning is useful, I think for most use cases using command line flags like -W:deprecated/-W:no-deprecated would be better
> (e.g., it's likely that some warnings are missed when doing Warning.enable(:deprecated) at runtime).

The Ruby-level API is intended to be used for test of warnings.
See assert_warn in test/ruby/test_keyword.rb and
tool/lib/test/unit/core_assertions.rb.

If we don't have such API, we need a process for each assertion for the warnings.

#9 - 11/13/2019 01:37 AM - shyouhei (Shyouhei Urabe)

Re: "Don't emit deprecation warnings by default" part of this proposal

I'm for this. The idea is that while we (ruby-core devs) want people to write new codes, that is not always possible by them... There are gems.

Today, it is almost always the case that a ruby application consists of many 3rd party gems which are out of control of the author of the application. When a new ruby version issues millions of warnings inside of a gem because the code inside is "deprecated", that is a very huge drawback for the application author to move to new ruby. The gem is (despite old) proven to work. The new ruby version is just adding annoyance, and not field-proven. There is no reason to move. THIS IS VERY BAD.

Of course, the deprecated features would disappear someday. Application authors have to brace the impact. ONLY WHEN they are ready to do so, the deprecation warnings then make sense. So the timing those warnings are useful depends on each situation, there should be ways to control enable/disabling them.

#10 - 11/13/2019 04:53 AM - sam.saffron (Sam Saffron)

I also support this, seems like a much better default. In fact when 3.0 upgrade becomes a must cause 2.7 is going EOL or something we could flip the default, but that is years out.

For context per: https://bugs.ruby-lang.org/issues/16289#change-82607

Running the current discourse test suite causes 2,698,774 lines to be written to STDERR. This will be very very welcome change.

#11 - 11/13/2019 06:31 AM - jeremyevans0 (Jeremy Evans)

I'm fine with Warning.disable(:deprecated) and similar methods. Those are nice as they open the door to disabling other warnings, such as method redefinition and uninitialized instance variables.

I'm against not emitting deprecation warnings by default. That makes them borderline useless, and not worth the effort to implement them at that point.

What people should understand is that at least for keyword arguments (not sure about other deprecation), Ruby goes out of the way to notify you for all places where behavior will change in 3.0 (or whatever version we decide to make the change in). A huge portion of the difficulty in implementing the keyword argument changes was for proper deprecation warnings. Not displaying them by default is wasting a huge amount of effort.

If an application generates 2 million warnings on Ruby 2.7, that's a huge indication that it needs to be updated, and should not be run in production until it has been updated so it runs without warnings. If you are running an old/stable gem and it implements any deprecation warnings, those are indications that it needs to be fixed and it should not be run in production on Ruby 2.7 until the warnings are fixed.

I think we should have the warnings on by default, and be very visible, so the problems they identify get fixed. Basically, the current behavior is best. It's better to fix the problems sooner than later. If the warnings are hidden, the problems will take longer to fix, and that hurts the community more than it helps, IMO. We should push developers and gem maintainers to update their code sooner rather than later.

#12 - 11/13/2019 07:13 AM - shyouhei (Shyouhei Urabe)

jeremyevans0 (Jeremy Evans) wrote:

> If an application generates 2 million warnings on Ruby 2.7, that's a huge indication that it needs to be updated, and should not be run in production until it has been updated so it runs without warnings. If you are running an old/stable gem and it implements any deprecation warnings, those are indications that it needs to be fixed and it should not be run in production on Ruby 2.7 until the warnings are fixed.

I cannot agree with this paragraph. This is not how backwards compatibility works. Running existing codes without any problem is a must.

#13 - 11/13/2019 07:25 AM - duerst (Martin Dürst)

- Related to Feature #16289: Reduce duplicated warnings for the change of Ruby 3 keyword arguments added

#14 - 11/13/2019 07:28 AM - akr (Akira Tanaka)

I found Python suppress deprecation warnings by default.

https://docs.python.org/3/library/warnings.html#warning-categories
https://docs.python.org/3/library/warnings.html#default-warning-filter

#15 - 11/13/2019 07:30 AM - duerst (Martin Dürst)

I agree that issuing 2 million warnings is easily too much. On the other hand I also agree that by default not issuing any warnings at all clearly seems way too few.

A good default should be somewhere between a single warning and a few warnings (or a warning count) by code location. For details, I refer to Feature #16289.

A single warning might look something like: "Warning: Using functionality that is deprecated. Please use ... to see more details and fix them.".

Settings would have to be extended to allow distinction between these few cases. Implementation may take some work, but I don't think it would be too difficult.

#16 - 11/13/2019 07:47 AM - duerst (Martin Dürst)

akr (Akira Tanaka) wrote:

> I found Python suppress deprecation warnings by default.
>
> https://docs.python.org/3/library/warnings.html#warning-categories
> https://docs.python.org/3/library/warnings.html#default-warning-filter

Please note the following at the second location:
"Changed in version 3.7: DeprecationWarning is once again shown by default when triggered directly by code in **main**."

So what Python currently does in issue depreciation warnings in the main program, but not in libraries and similar code. That may also be a reasonable way to limit the number of warnings while making sure deprecations don't go unnoticed (because that makes them useless).

#17 - 11/13/2019 08:09 AM - akr (Akira Tanaka)

duerst (Martin Dürst) wrote:

> akr (Akira Tanaka) wrote:
>
> > I found Python suppress deprecation warnings by default.
> >
> > https://docs.python.org/3/library/warnings.html#warning-categories
> > https://docs.python.org/3/library/warnings.html#default-warning-filter
>
> Please note the following at the second location:
> "Changed in version 3.7: DeprecationWarning is once again shown by default when triggered directly by code in **main**."
>
> So what Python currently does in issue depreciation warnings in the main program, but not in libraries and similar code. That may also be a reasonable way to limit the number of warnings while making sure deprecations don't go unnoticed (because that makes them useless).

Thank you for pointing that.

I think it's possible compromise in Ruby.

Although Ruby has no generic warning mechanism as Python,
I think hard-coding the condition
(show deprecation warnings only in main file by default)
is not difficult.

#18 - 11/13/2019 08:38 AM - duerst (Martin Dürst)

akr (Akira Tanaka) wrote:

> Although Ruby has no generic warning mechanism as Python,

Indeed Ruby doesn't have such a mechanism, but with this discussion we are quickly moving in the direction of introducing such a mechanism, and should be aware of that fact.

#19 - 11/13/2019 09:35 AM - nobu (Nobuyoshi Nakada)

Have I shown this yesterday? https://github.com/nobu/ruby/pull/new/feature/Warning.warn-category

#20 - 11/13/2019 09:46 AM - akr (Akira Tanaka)

nobu (Nobuyoshi Nakada) wrote:

> Have I shown this yesterday? https://github.com/nobu/ruby/pull/new/feature/Warning.warn-category

Pull request needs GitHub login.
Compare is better.

https://github.com/ruby/ruby/compare/master...nobu:feature/Warning.warn-category

#21 - 11/13/2019 09:02 PM - sam.saffron (Sam Saffron)

> I'm against not emitting deprecation warnings by default.

I disagree with this, in this case the enormous amount of complaining hurts the ecosystem applies unneeded urgency on people and causes conflict in the community.

We have been through this previously with secret ENV vars that make Ruby "faster". Forcing people to learn about a magical env var to disable deprecations they have no control over is in the same boat. Besides, this will just end up being reported as a security bug to the security list if left as is, cause docker will eat up all your disk space due to a single call site flooding STDERR forcing logs to grow forever.

Fixing your own app may be straightforward, in the Discourse case it was just editing a few files, but forcing people to start piling on and pressuring gem authors cause 2.7 is out and now and the gem they released is causing a big pile of warnings and they happen to be on a 3 week break and now they need to hunt down a computer and commit a complex patch cause delegation is now very messy, would not be nice.

We have at least a year here to clean up the mess after 2.7 is released, we don't need an artificial urgency here. I am happy to strive to run Discourse deprecation free as soon as possible, but if a gem author takes 2 months to sort this out this is not a huge deal for me.

If we must an alternative here is to release a minor of 2.7 say 9 months from now that flicks the default cause the fire is imminent.

#22 - 11/14/2019 07:41 AM - jeremyevans0 (Jeremy Evans)

sam.saffron (Sam Saffron) wrote:

> > I'm against not emitting deprecation warnings by default.
>
> I disagree with this, in this case the enormous amount of complaining hurts the ecosystem applies unneeded urgency on people and causes conflict in the community.

I'm OK with eliminating duplicate warnings, or stopping warning after a certain number of warnings emitted. Not necessarily in favor of either, but either is far superior to not warning at all by default.

> We have been through this previously with secret ENV vars that make Ruby "faster". Forcing people to learn about a magical env var to disable deprecations they have no control over is in the same boat. Besides, this will just end up being reported as a security bug to the security list if left as is, cause docker will eat up all your disk space due to a single call site flooding STDERR forcing logs to grow forever.

Again, this problem better fixed by eliminating duplicate warnings or stopping after a certain number of warnings, compared to not warning at all by default.

> Fixing your own app may be straightforward, in the Discourse case it was just editing a few files, but forcing people to start piling on and pressuring gem authors cause 2.7 is out and now and the gem they released is causing a big pile of warnings and they happen to be on a 3 week break and now they need to hunt down a computer and commit a complex patch cause delegation is now very messy, would not be nice.

There should be pressure on maintainers of popular gems to make sure they continue to work on newer versions of Ruby. However, there is no urgent need. Gem maintainers have been able to test their code with the master branch for a couple months now and fix possible issues, and preview2 for a few weeks. If their gem isn't ready for 2.7 before the release of 2.7 final, then there should be no problems with users of the gems continuing to run 2.6 until the gem has been updated. If they want to use the gem with Ruby 2.7 before the gem has been updated to fix the warnings in 2.7, they should be OK with disabling the warnings manually.

> We have at least a year here to clean up the mess after 2.7 is released, we don't need an artificial urgency here. I am happy to strive to run Discourse deprecation free as soon as possible, but if a gem author takes 2 months to sort this out this is not a huge deal for me.

It's not a huge deal to me either. However, it also isn't a huge deal if they have to run Discourse on 2.6 instead of 2.7 until Discourse and Discourse's dependencies are updated. If they want to use it before then, they can add a line of code to disable the warnings if they don't care.

> If we must an alternative here is to release a minor of 2.7 say 9 months from now that flicks the default cause the fire is imminent.

Minor releases of Ruby should only be done to fix bugs, they should not make any feature changes, in my opinion.

I am not one to try to predict the future, but I think it is very likely if warnings are disabled by default in 2.7, and the changes take effect in 3.0, there will be a huge backlash of people complaining that we broke things without proper deprecation warnings when 3.0 is released. I guess if we'd prefer a huge backlash then than now, disabling the warnings could make sense. However, I'd rather have annoying warnings now than broken code later.

shyouhei (Shyouhei Urabe) wrote:

> jeremyevans0 (Jeremy Evans) wrote:
>
> > If an application generates 2 million warnings on Ruby 2.7, that's a huge indication that it needs to be updated, and should not be run in production until it has been updated so it runs without warnings. If you are running an old/stable gem and it implements any deprecation warnings, those are indications that it needs to be fixed and it should not be run in production on Ruby 2.7 until the warnings are fixed.
>
> I cannot agree with this paragraph. This is not how backwards compatibility works. Running existing codes without any problem is a must.

The logical conclusion of this reasoning is that no deprecation warnings should ever be added, since any deprecation warning could theoretically cause a problem. However, I don't think that is a good approach. What are your opinions of not warning at all by default compared to eliminating duplicate warnings or stopping after a certain number of warnings?

#23 - 11/15/2019 01:43 AM - shyouhei (Shyouhei Urabe)

In short: yes, warn iff not seen before can be an acceptable option.

jeremyevans0 (Jeremy Evans) wrote:

> The logical conclusion of this reasoning is that no deprecation warnings should ever be added, since any deprecation warning could theoretically cause a problem.

Well if you go extremely strict, you are right. Any warnings could theoretically cause problems. And yet, they are OK if explicitly enabled at runtime. There is a clear intention that a user needs those warnings then. Not emitting deprecation warnings "by default" is a win.

In practice -- backward compatibility issues are always practices -- what is a "problem" and what is not is very fuzzy. Millions of lines of warnings flooding the STDERR is clearly troublesome. But what about only one warning at bootup? Obviously less. Maybe acceptable. Then what about dozens? Or hundreds? Or thousands?

Also, because warnings go through Warning.warn, it involves Ruby-level string allocation every time. Logging a warning to storages involves IO. These overheads are hard to be admitted during accepting HTTP requests, but maybe not that severe when they show up at bootup and keep silent for the rest of a process lifetime.

> However, I don't think that is a good approach. What are your opinions of not warning at all by default compared to eliminating duplicate warnings or stopping after a certain number of warnings?

So what I think is wrong is those warnings continuously displayed. I prefer the way requested in this ticket, but #16289 is practically an acceptable solution to me.

#24 - 11/15/2019 02:28 AM - jeremyevans0 (Jeremy Evans)

shyouhei (Shyouhei Urabe) wrote:

> In short: yes, warn iff not seen before can be an acceptable option.

I strongly urge the approach taken in #16289 then. It seems to be the best compromise that still notifies users about problems that need to be fixed, without causing so many warnings as to present usability problems.

sam.saffron (Sam Saffron) any chance you could try https://github.com/ruby/ruby/pull/2458 and let us know how many warnings are reported for Discourse? That would allow us to see if it could be an acceptable compromise for large applications.

We could still keep the proposed Warning.enable and Warning.disable methods in this ticket to control whether warnings are emitted.

In addition to the keyword argument separation changes, the other major deprecation warnings in 2.7 will be the ones for $SAFE/taint.

#25 - 11/15/2019 03:02 AM - akr (Akira Tanaka)

Dan0042 (Daniel DeLorme) wrote:

> Personally I'd like to have this feature as an attribute accessor of Warning. So it's easier to set the value based on configuration.
>
> ```
> Warning[:deprecated] = true
> ```

```
Warning[:deprecated] = ENV.include?('WARN_DEPRECATED')
```

Apart from environment variable based configuration,
such method would be useful for restoring configuration.

```
begin
  w = Warning[:deprecated]
  Warning[:deprecated] = false
  ...something...
ensure
  Warning[:deprecated] = w
end
```

is shorter than

```
begin
  w = Warning.enabled?(:deprecated)
  Warning.disable(:deprecated)
  ...something...
ensure
  if w then Warning.enable(:deprecated) else Warning.disable(:deprecated) end
end
```

#26 - 11/15/2019 05:16 PM - Eregon (Benoit Daloze)

shyouhei (Shyouhei Urabe) wrote:

> I prefer the way requested in this ticket, but #16289 is practically an acceptable solution to me.

Then let's do #16289 then, I agree that's a good solution to mitigate too many warnings.

#27 - 11/16/2019 12:46 AM - sam.saffron (Sam Saffron)

> any chance you could try https://github.com/ruby/ruby/pull/2458 and let us know how many warnings are reported for Discourse?

Sure! will give it a shot next week, also want to double check we don't see any noticeable perf impact for all the accounting.

#28 - 11/16/2019 03:49 PM - Eregon (Benoit Daloze)

sam.saffron (Sam Saffron) wrote:

> Sure! will give it a shot next week, also want to double check we don't see any noticeable perf impact for all the accounting.

It's actually a lot faster with deduplicated warnings, see my comment on that issue.

#29 - 11/16/2019 10:58 PM - sam.saffron (Sam Saffron)

> It's actually a lot faster

A lot faster than no warning at all? Was just concerned looking up the callsite and a hashtable lookup might have some sort of cost. Also this technically will leak memory for the lifetime of the process. Fancy meta programming can blow memory.

#30 - 11/17/2019 11:14 AM - Eregon (Benoit Daloze)

sam.saffron (Sam Saffron) wrote:

> A lot faster than no warning at all? Was just concerned looking up the callsite and a hashtable lookup might have some sort of cost. Also this technically will leak memory for the lifetime of the process. Fancy meta programming can blow memory.

Faster than a warning printed on every call. I would think the lookup only happens if it would warn, and that the lookup is faster than printing: https://bugs.ruby-lang.org/issues/16289#note-5

#31 - 11/26/2019 04:22 PM - akr (Akira Tanaka)

I still think disabling deprecation warnings is better default.

The warnings are useless for end users of applications written in Ruby.

Consider Ruby 2.7 will be distributed by an OS (such as Debian) and
some gems are not deprecation-warning free.
I expect there are some gems are not updated before a release of the OS.
So, a user of application written in Ruby on such OS will see deprecation warnings
even if the user doesn't know Ruby.
I think it is frustrating situation.

Also, the warnings may be difficult to remove because
such OS may slow to update gems because stability.
This doesn't cause actual problems because the OS controls Ruby version and
applications should work well on Ruby 2.7.
The important thing is working application,
not removing deprecation warnings.

I agree that disabling deprecation warnings by default makes
less pressure for developers.

However there are several situations that
we can know a user is a developer (who write Ruby code):

- unit test
- irb

In such situation, we can emit deprecation warnings
for developers without frustrating end users.

Note that reducing duplicated warnings is good thing
for developers.
But I don't think it's enough for end users.

#32 - 11/27/2019 07:20 AM - sam.saffron (Sam Saffron)

I agree 100% with akr here.

Even with Eregon (Benoit Daloze)'s proposed patch running our test suite with warnings is unusable cause of pages of:

```
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:29: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:31: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:34: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
```

```
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning: 
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning: 
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning: 
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning: 
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_controller/metal/request_
forgery_protection.rb:285: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning: 
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning: 
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:29: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:31: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:34: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
```

```
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:29: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:31: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:34: warning: rb_safe_
level will be removed in Ruby 3.0
```

I know Jeremy is not keen on this, but tracking callsites is a memory leak and I guess if you wanted to twist my arm hard I could go with warn on first instance EVER.

```
Warning.enable(:deprecated, :once)
```

Where this means we only warn one time per "type" of deprecation, it outputs to STDERR then sets a bool to say, don't warn again from here. Technically with "first time ever" a developer could still pick up on every deprecation.

#33 - 11/27/2019 02:01 PM - mame (Yusuke Endoh)

When we consider the default verbosity of warnings, we should care about the "audience" of the warnings.

In the previous developers' meeting, the default warnings should target server operators (such as a SRE team) as a primary audience. We thought that they don't want to see deprecation warnings, so off-by-default seemed good.
However, after the meeting, I listened opinions in my company, and one of them said: "We don't care the default verbosity. Regardless of this paticular issue, we will have to change some pieces of code to upgrade the Ruby version. So, if we can stop the warnings by adding a small code change, it would be good enough."

At the present time, I think the default warnings should target developres. The verbosity should be decided based on importance, i.e., on-by-default for near-future deprecation warnings.

We already have a way to stop all warnings: $VERBOSE = nil. More flexible mechanism to control warning verbosity based on category would be good-to-have. However, there is no time for 2.7. I'm unsure if we should hurry to introduce the mechanism.

#34 - 11/27/2019 06:20 PM - jeremyevans0 (Jeremy Evans)

sam.saffron (Sam Saffron) wrote:

> I agree 100% with akr here.
>
> Even with Eregon (Benoit Daloze)'s proposed patch running our test suite with warnings is unusable cause of pages of:

Note that the vast majority of the warnings listed are not keyword argument separation warnings, they are safe level and String#match? warnings. I think the patch you are using only handles keyword argument separation warnings. If you extended the approach to other deprecation warnings, which would dedup the warnings, you end up with:

```
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_controller/metal/request_
forgery_protection.rb:285: warning: given argument is nil; this will raise a Typ
eError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a
 TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:29: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:31: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:34: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
rb_safe_level will be removed in Ruby 3.0
```

That looks pretty manageable, don't you think? I think that list is far better than hiding by default.

> I know Jeremy is not keen on this, but tracking callsites is a memory leak and I guess if you wanted to twist my arm hard I could go with warn on first instance EVER.

Tracking callsites can be a memory leak if your callsites are transient (i.e. there is no fixed bound to them). That's an unusual case, though. Do you have an example of a Ruby application where this would cause an actual memory leak (unbounded memory growth)?

> `Warning.enable(:deprecated, :once)`
>
> Where this means we only warn one time per "type" of deprecation, it outputs to STDERR then sets a bool to say, don't warn again from here. Technically with "first time ever" a developer could still pick up on every deprecation.

True, it just takes more test runs, as each test run will only uncover a single issue. I'm not against :once reporting as an option, but I don't think it is a good default.

akr (Akira Tanaka) wrote:

> I still think disabling deprecation warnings is better default.
>
> The warnings are useless for end users of applications written in Ruby.
>
> Consider Ruby 2.7 will be distributed by an OS (such as Debian) and
> some gems are not deprecation-warning free.
> I expect there are some gems are not updated before a release of the OS.
> So, a user of application written in Ruby on such OS will see deprecation warnings
> even if the user doesn't know Ruby.
> I think it is frustrating situation.

Agreed, warning messages can be frustrating. However, it is much more frustrating if something stops working completely, which is the likely outcome in Ruby 3.0 of hiding deprecation warnings by default in 2.7.

> Also, the warnings may be difficult to remove because
> such OS may slow to update gems because stability.
> This doesn't cause actual problems because the OS controls Ruby version and
> applications should work well on Ruby 2.7.
> The important thing is working application,
> not removing deprecation warnings.
>
> I agree that disabling deprecation warnings by default makes
> less pressure for developers.

Disabling deprecation warnings by default just delays the pressure till 3.0. Then the pressure will be even greater, since instead of just displaying warnings, the software will stop working completely.

> However there are several situations that
> we can know a user is a developer (who write Ruby code):
>
> - unit test
> - irb

If irb is used, we can know that and change warning behavior. However, many developers use pry, so changing warning behavior just in irb is not a solution as those developers will not see the warnings.

Likewise, if a test library from stdlib is used, we could know that and change warning behavior. However, many developers use other test libraries, so changing warning behavior just in test libraries in stdlib is not a solution as those developers will not see the warnings.

#35 - 11/27/2019 08:29 PM - Eregon (Benoit Daloze)

sam.saffron (Sam Saffron) wrote:

> Even with Eregon (Benoit Daloze)'s proposed patch running our test suite with warnings is unusable cause of pages of:

Do you mean mame (Yusuke Endoh)'s PR from #16289, that is https://github.com/ruby/ruby/pull/2458 ?

Could you try with Jeremy's approach in https://github.com/ruby/ruby/pull/2458#issuecomment-531269374:

```
WARNINGS_SEEN = {}
def Warning.warn(str)
  unless WARNINGS_SEEN[str]
    WARNINGS_SEEN[str] = true
    super
  end
end
```

That approach should filter every duplicate warning.

If it's still too much, I guess a limit of e.g., 100 warnings might be a way to make it more reasonable (and also print warnings are omitted after 100 of them were printed).
A limit of 1 would be very inconvenient to work with when fixing the cause warnings, and not give an idea if an app still has many warnings or not.

#36 - 11/27/2019 10:54 PM - akr (Akira Tanaka)

jeremyevans0 (Jeremy Evans) wrote:

> > However there are several situations that
> > we can know a user is a developer (who write Ruby code):
> >
> > - unit test

- irb

  If irb is used, we can know that and change warning behavior. However, many developers use pry, so changing warning behavior just in irb is not a solution as those developers will not see the warnings.

  Likewise, if a test library from stdlib is used, we could know that and change warning behavior. However, many developers use other test libraries, so changing warning behavior just in test libraries in stdlib is not a solution as those developers will not see the warnings.

Of course, we can enable deprecation warnings for
pry and other test libraries.

#37 - 11/27/2019 11:13 PM - sam.saffron (Sam Saffron)

I tried Jeremy's patch results are here of running Discourse bin/turbo_rspec it runs our test suite using 16 threads on my computer.

2440 warning for every test run, suffice to say at Discourse we will absolutely turn warnings off after submitting tickets and waiting for them to be handled. Seeing anything that is not purposeful in the test runners logs interferes greatly with puts debugging.

https://gist.github.com/SamSaffron/724f821df9786a2ddd408202b950cb46

In a multithreaded setup "warn per thing" and even warn "once per type" and "warn by callsite" are still too noisy. In a single threaded app I can see "warn per thing" as useful sometimes.

A side effect of all this tracking is that there is a small perf penalty over doing no work at all. (3 million hash lookups)

As for what Discourse will do regardless of what Ruby decide:

- We will amend our default for quality of life for dev team not to warn by default unless you opt in via env, this sucks but we are going to simply run def Warning.warn(str); end this will be the case till the storm subsides, once we get it to 0 we will re-enable warnings.
- In production we will not be warning on deprecations ever, due to risk of flooding logs so the same patch will apply, but it is permanent. Prob just add RUBYOPT=-W0 to env for production for risk mitigation.

Technically there is nothing that is stopping us moving in this way when 2.7 is released.

I am curious at what GitHub and Shopify will do here who both have reasonably big dev teams?

I do worry that the current state is dangerous:

- When you upgrade your Ruby you can be presented with 3 million lines in STDERR for a process that runs for a couple of minutes it will be reported to the security mailing list as a flaw.
- I worry that a huge increase in warnings will place enormous urgent pressure on gem maintainers.

Overall this is a transient state so I defer to Matz on how to deal with this, as it stands if we ship 2.7 as it I am counting down 2 days before this is submitted to the security mailing list as a security flaw.

#38 - 11/27/2019 11:30 PM - Eregon (Benoit Daloze)

It's actually only 130 unique warnings:

```
$ wc -l run.txt
2439 run.txt
$ grep -v 'Setting up parallel test mode - starting' run.txt > run2.txt
$ wc -l run2.txt
2408 run2.txt
$ cat run2.txt | sort | uniq | wc -l
130
```

This shows the approach above with a Ruby Hash doesn't cut it with concurrent threads, but we could fix that with some API on Warning which does the proper synchronization and warn if put-if-absent.
And probably just having that as the default behavior would be good.

I can actually copy-paste the list of unique warnings here.
It seems all things that are likely to break when migrating in 3.0, so the warnings seem important for gems and apps to fix them before migrating to 3.0.

```
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/abstract_controller/helpers.rb:6
7: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_controller/metal/request_
forgery_protection.rb:285: warning: given argument is nil; this will raise a TypeError in the next release
```

```
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_controller/metal/request_
forgery_protection.rb:313: warning: for `form_authenticity_token' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/middleware/cooki
es.rb:635: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/middleware/stack
.rb:37: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/middleware/stati
c.rb:110: warning: for `initialize' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/routing/mapper.r
b:344: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/routing/mapper.r
b:353: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/testing/integrat
ion.rb:17: warning: for `get' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/testing/integrat
ion.rb:23: warning: for `post' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/testing/integrat
ion.rb:357: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/testing/integrat
ion.rb:35: warning: for `put' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionpack-6.0.1/lib/action_dispatch/testing/integrat
ion.rb:41: warning: for `delete' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/helpers/tag_helper.r
b:111: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/helpers/tag_helper.r
b:44: warning: for `tag_string' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/helpers/translation_
helper.rb:120: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/lookup_context.rb:14
0: warning: for `template_exists?' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/renderer/abstract_re
nderer.rb:20: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/template.rb:130: war
ning: for `initialize' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/unbound_template.rb:
24: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/actionview-6.0.1/lib/action_view/view_paths.rb:11: wa
rning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activemodel-6.0.1/lib/active_model/attribute_mutation
_tracker.rb:167: warning: for `changed?' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activemodel-6.0.1/lib/active_model/attribute_mutation
_tracker.rb:45: warning: for `changed?' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activemodel-6.0.1/lib/active_model/callbacks.rb:145:
warning: for method defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activemodel-6.0.1/lib/active_model/dirty.rb:170: warn
ing: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activemodel-6.0.1/lib/active_model/type/integer.rb:13
: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activemodel-6.0.1/lib/active_model/type/value.rb:8: w
arning: for `initialize' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/associations.rb:
1370: warning: for `has_many' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/associations.rb:
1860: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/attribute_method
s/dirty.rb:102: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/attribute_method
s/dirty.rb:52: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/callbacks.rb:318
: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/abstract_adapter.rb:90: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/abstract/database_statements.rb:274: warning: for `transaction' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/abstract/transaction.rb:145: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/abstract/transaction.rb:78: warning: for `initialize' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql_adapter.rb:620: warning: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/connection_adapt
ers/postgresql/oid/specialized_string.rb:12: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/persistence.rb:4
70: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/persistence.rb:5
```

```
03: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/persistence.rb:8
51: warning: for `touch' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/railties/collect
ion_cache_association_loading.rb:12: warning: for `relation_from_options' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/railties/collect
ion_cache_association_loading.rb:7: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/relation/delegat
ion.rb:115: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/relation.rb:27:
warning: for `initialize' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/timestamp.rb:127
: warning: for `create_or_update' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/transactions.rb:
212: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/type/adapter_spe
cific_registry.rb:9: warning: for `add_modifier' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activerecord-6.0.1/lib/active_record/type.rb:27: warn
ing: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activesupport-6.0.1/lib/active_support/inflector/meth
ods.rb:128: warning: for `humanize_without_cache' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activesupport-6.0.1/lib/active_support/inflector/meth
ods.rb:174: warning: for `titleize_without_cache' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activesupport-6.0.1/lib/active_support/message_encryp
tor.rb:150: warning: for `encrypt_and_sign' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activesupport-6.0.1/lib/active_support/message_encryp
tor.rb:175: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/activesupport-6.0.1/lib/active_support/messages/metad
ata.rb:17: warning: for `wrap' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/aws-sdk-core-3.48.6/lib/seahorse/client/configuration
.rb:107: warning: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/aws-sdk-core-3.48.6/lib/seahorse/client/http/response
.rb:125: warning: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/aws-sdk-core-3.48.6/lib/seahorse/client/http/response
.rb:133: warning: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/aws-sdk-core-3.48.6/lib/seahorse/client/plugin.rb:61:
 warning: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/excon-0.64.0/lib/excon.rb:178: warning: Capturing the
 given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/faraday-0.15.4/lib/faraday/options.rb:166: warning: C
apturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/ffi-1.10.0/lib/ffi/pointer.rb:97: warning: rb_safe_le
vel will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/highline-1.7.10/lib/highline.rb:624: warning: The las
t argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/i18n-1.7.0/lib/i18n.rb:279: warning: for `localize' d
efined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/mini_suffix-0.3.0/lib/mini_suffix.rb:23: warning: rb_
safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/multi_json-1.13.1/lib/multi_json/options_cache.rb:12:
 warning: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/onebox-1.9.23/lib/onebox/engine/json.rb:9: warning: c
alling URI.open via Kernel#open is deprecated, call URI.open directly
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/onebox-1.9.23/lib/onebox/helpers.rb:222: warning: URI
.escape is obsolete
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/rotp-3.3.1/lib/rotp/totp.rb:85: warning: URI.escape i
s obsolete
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/rotp-3.3.1/lib/rotp/totp.rb:93: warning: URI.escape i
s obsolete
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/rspec-core-3.8.0/lib/rspec/core/metadata.rb:172: warn
ing: deprecated Object#=~ is called on Class; it always returns nil
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:29: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:31: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/engine.rb:34: warning: rb_safe_
level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/functions_handler.rb:33: warnin
g: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/import_handler.rb:43: warning:
 rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/script/value_conversion/string.
rb:9: warning: rb_safe_level will be removed in Ruby 3.0
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/script/value/string.rb:26: warn
ing: deprecated Object#=~ is called on FalseClass; it always returns nil
```

```
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/script/value/string.rb:26: warn
ing: deprecated Object#=~ is called on Float; it always returns nil
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/script/value/string.rb:26: warn
ing: deprecated Object#=~ is called on Integer; it always returns nil
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sassc-2.0.1/lib/sassc/script/value/string.rb:26: warn
ing: deprecated Object#=~ is called on TrueClass; it always returns nil
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/sprockets-3.7.2/lib/sprockets/http_utils.rb:46: warni
ng: given argument is nil; this will raise a TypeError in the next release
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/test-prof-0.9.0/lib/test_prof/before_all.rb:77: warni
ng: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/test-prof-0.9.0/lib/test_prof/event_prof/custom_event
s.rb:10: warning: Capturing the given block using Proc.new is deprecated; use `&block` instead
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/test-prof-0.9.0/lib/test_prof/recipes/rspec/let_it_be
.rb:49: warning: The last argument is used as the keyword parameter
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/test-prof-0.9.0/lib/test_prof/recipes/rspec/let_it_be
.rb:53: warning: for `let_it_be' defined here
/home/sam/.rbenv/versions/trunk/lib/ruby/gems/2.7.0/gems/tzinfo-1.2.5/lib/tzinfo/ruby_core_support.rb:142: war
ning: The last argument is used as the keyword parameter
/home/sam/Source/discourse/app/controllers/list_controller.rb:377: warning: URI.escape is obsolete
/home/sam/Source/discourse/app/controllers/metadata_controller.rb:28: warning: given argument is nil; this wil
l raise a TypeError in the next release
/home/sam/Source/discourse/app/controllers/reviewables_controller.rb:36: warning: The last argument is used as
 the keyword parameter
/home/sam/Source/discourse/app/controllers/reviewables_controller.rb:37: warning: The last argument is used as
 the keyword parameter
/home/sam/Source/discourse/app/controllers/topics_controller.rb:449: warning: The last argument is used as the
 keyword parameter
/home/sam/Source/discourse/app/mailers/user_notifications.rb:627: warning: URI.escape is obsolete
/home/sam/Source/discourse/app/mailers/user_notifications.rb:652: warning: URI.escape is obsolete
/home/sam/Source/discourse/app/models/concerns/has_url.rb:25: warning: URI.unescape is obsolete
/home/sam/Source/discourse/app/models/embeddable_host.rb:37: warning: URI.unescape is obsolete
/home/sam/Source/discourse/app/models/post.rb:651: warning: for `rebake!' defined here
/home/sam/Source/discourse/app/models/post.rb:974: warning: URI.unescape is obsolete
/home/sam/Source/discourse/app/models/remote_theme.rb:149: warning: The last argument is used as the keyword p
arameter
/home/sam/Source/discourse/app/models/reviewable.rb:411: warning: for `list_for' defined here
/home/sam/Source/discourse/app/models/tag.rb:75: warning: for `top_tags' defined here
/home/sam/Source/discourse/app/models/theme.rb:341: warning: for `set_field' defined here
/home/sam/Source/discourse/app/models/topic_list.rb:65: warning: The last argument is used as the keyword para
meter
/home/sam/Source/discourse/app/models/topic.rb:1113: warning: for `set_or_create_timer' defined here
/home/sam/Source/discourse/app/models/topic.rb:1354: warning: URI.escape is obsolete
/home/sam/Source/discourse/app/models/user.rb:201: warning: deprecated Object#=~ is called on Array; it always
 returns nil
/home/sam/Source/discourse/lib/auth/open_id_authenticator.rb:39: warning: for `after_authenticate' defined her
e
/home/sam/Source/discourse/lib/compression/engine.rb:25: warning: The last argument is used as the keyword par
ameter
/home/sam/Source/discourse/lib/compression/strategy.rb:35: warning: for `strip_directory' defined here
/home/sam/Source/discourse/lib/email/sender.rb:167: warning: deprecated Object#=~ is called on Mail::Field; it
 always returns nil
/home/sam/Source/discourse/lib/freedom_patches/inflector_backport.rb:30: warning: The last argument is used as
 the keyword parameter
/home/sam/Source/discourse/lib/plugin/instance.rb:214: warning: The last argument is used as the keyword param
eter
/home/sam/Source/discourse/lib/seed_data/categories.rb:148: warning: for `update_category' defined here
/home/sam/Source/discourse/lib/seed_data/categories.rb:15: warning: The last argument is used as the keyword p
arameter
/home/sam/Source/discourse/lib/seed_data/categories.rb:24: warning: The last argument is used as the keyword p
arameter
/home/sam/Source/discourse/lib/seed_data/categories.rb:98: warning: for `create_category' defined here
/home/sam/Source/discourse/lib/seed_data/topics.rb:126: warning: for `create_topic' defined here
/home/sam/Source/discourse/lib/seed_data/topics.rb:152: warning: for `update_topic' defined here
/home/sam/Source/discourse/lib/seed_data/topics.rb:16: warning: The last argument is used as the keyword param
eter
/home/sam/Source/discourse/lib/seed_data/topics.rb:26: warning: The last argument is used as the keyword param
eter
/home/sam/Source/discourse/lib/site_setting_extension.rb:387: warning: The last argument is used as the keywor
d parameter
/home/sam/Source/discourse/lib/site_settings/deprecated_settings.rb:66: warning: for method defined here
/home/sam/Source/discourse/lib/tasks/posts.rake:140: warning: The last argument is used as the keyword paramet
er
/home/sam/Source/discourse/lib/url_helper.rb:13: warning: URI.escape is obsolete
/home/sam/Source/discourse/lib/url_helper.rb:51: warning: URI.escape is obsolete
/home/sam/Source/discourse/lib/user_name_suggester.rb:14: warning: deprecated Object#=~ is called on Integer;
```

```
it always returns nil
/home/sam/Source/discourse/lib/validators/url_validator.rb:12: warning: URI.escape is obsolete
/home/sam/Source/discourse/spec/components/auth/open_id_authenticator_spec.rb:12: warning: The keyword argumen
t is passed as the last hash parameter
/home/sam/Source/discourse/spec/components/auth/open_id_authenticator_spec.rb:19: warning: The keyword argumen
t is passed as the last hash parameter
/home/sam/Source/discourse/spec/requests/list_controller_spec.rb:184: warning: URI.escape is obsolete
Warning: no type cast defined for type "name" with oid 19. Please cast this type explicitly to TEXT to be safe
 for future changes.
```

#39 - 11/27/2019 11:34 PM - Eregon (Benoit Daloze)

The above also shows we should avoid multiple calls to rb_warn* for a single logical warning.
E.g., warning: The last argument is used as the keyword parameter and warning: for method defined here should be part of a single (multi-line) String sent to Warning.warn.
That's anyway much better for other kinds of filtering with Warning.warn.

#40 - 11/28/2019 05:44 AM - matz (Yukihiro Matsumoto)

By default, it should emit (deprecation) warnings. It's a good idea to have an environment variable option to suppress deprecation warnings.

Matz.

#41 - 11/29/2019 01:46 AM - sam.saffron (Sam Saffron)

Matz,

We already have RUBYOPT=-W0 so no change is needed here for a global suppression. Maybe add RUBYOPT=--no-deprecation or something like that.

The question is about the default here:

1. Should default be "warn once per unique" deprecation: (parallel tests for Discourse mean 2440 lines to STDERR)
2. Should default be "warn every time" per deprecation: (parallel tests for Discourse mean 3 Million lines to STDERR)

#42 - 12/06/2019 01:33 PM - vo.x (Vit Ondruch)

- Related to Feature #16018: Add a way to deprecate methods added

#43 - 12/06/2019 01:34 PM - vo.x (Vit Ondruch)

In the context of this ticket, I have to reference my proposal #16018: "Add a way to deprecate methods"

#44 - 12/20/2019 02:19 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset git|a84ad24386d27269b90794146c2a351c1d79471b.

Added -W: command line option

To manage Warning[category] flags. Only -W:deprecated and
-W:no-deprecated are available now. [Feature #16345]

#45 - 07/16/2020 02:33 AM - akr (Akira Tanaka)

- Related to Feature #17000: 2.7.2 turns off deprecation warnings by default added

#46 - 07/20/2020 04:59 AM - matz (Yukihiro Matsumoto)

- Status changed from Closed to Open

In https://bugs.ruby-lang.org/issues/16345#note-40, I said it should be on by default, but it turned out they are too noisy.
So I changed my mind. Deprecation warnings should be turned on manually (probably with -Wdeprecate or Warning[:deprecated]=true).

Matz.

#47 - 07/21/2020 01:58 PM - akr (Akira Tanaka)

matz (Yukihiro Matsumoto) wrote in #note-46:

> So I changed my mind. Deprecation warnings should be turned on manually (probably with -Wdeprecate or Waning[:deprecated]=true).

I'm glad to hear that.

I still think deprecation warnings should not be emitted by default.

#48 - 07/27/2020 11:36 PM - mame (Yusuke Endoh)

I'm not sure if this is the right way.

I agree that a deprecation warning is annoying for end users. I also understand that application developers don't want them to be nervous. However, disabling the warnings by default does not solve the deprecation issue itself, but just postpone it. And the issue will be suddenly active when Ruby version is upgraded. Do people really want to see sudden break?

That being said, now I'm not strongly against the policy change.

- Some people actually complain the deprecation warnings for keyword change.
- The issue can be mitigated if all test frameworks enable all deprecation warnings.
- Sudden break is never a good thing, but may be a very strong motivation to fix applications (though it makes all users more nervous.)

But, I'm still unsure if it is a good practice in total.

#49 - 07/28/2020 12:29 AM - marcandre (Marc-Andre Lafortune)

Deprecation warnings for things that will break in the next version should be on by default (that policy being for Ruby or for gems). If that's too much warnings, then the breaking change should happen more slowly: first version with opt-in deprecation warnings, then version with opt-out deprecations, then breaking change.
It's way too late for that in late July for Ruby 2.7.

#50 - 07/30/2020 01:39 AM - ko1 (Koichi Sasada)

marcandre (Marc-Andre Lafortune) wrote in #note-49:

> If that's too much warnings, then the breaking change should happen more slowly: first version with opt-in deprecation warnings, then version with opt-out deprecations, then breaking change.

+1

Keyword warnings are suddenly introduced exceptionally.
I don't think we should choose the policy based on this exceptional case.

#51 - 08/24/2020 05:42 PM - akr (Akira Tanaka)

The enabled-by-default deprecation warnings have following effects.

- Good Effect: It makes developers more aware of the problem. (including users noticing and creating a issue)
- Bad Effect: Users ignore warnings or stop ruby upgrade

Since keyword warnings are so common, these effects are outstanding, but the both effects are not specific to keyword warnings.

We can avoid the bad effect and preserve the good effect by
disabled-by-default deprecation warnings and
changing developer's practice to enable deprecation warnings.

The developer's practice can be supported by tools,
such as test frameworks enable deprecation warnings automatically.

I think two-step migration,
disabled-by-default deprecation warnings and enabled-by-default deprecation warnings,
is not good idea.
It is important that developers can choose timing of dealing incompatibilities.
The enabled-by-default deprecation warning forces developers to deal it,
so it limits the timing.

#52 - 08/26/2020 06:02 AM - ko1 (Koichi Sasada)

We discussed about it and I recognized the motivation is how to shorten the migration period (to reuse conflicts feature or to delete barrier features) and I understand the advantages of this proposal.
IMO off-by-default and on-by-default should be discussed every time, but I don't against to make it as default policy.

#53 - 08/31/2020 05:36 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Closed

I agree with marcandre (Marc-Andre Lafortune) for making migration slow(er). We should wait for a few versions after emitting opt-in warnings before

removing the feature.
But it doesn't affect the discussion of stopping forced (default-on) warnings. I vote for making deprecation warnings basically opt-in.

Matz.