This patch introduces a new text formatting named **CommonMark Markdown (GitHub Flavored)**. It is based on [CommonMarker](https://github.com/ Corporation/CommonMarker) and [HTMLPipeline](https://github.com/ Corporation/HTMLPipeline). The formatter was extracted from Planio where it will soon become the default for new accounts.

We built this instead of going with the existing RedCarpet Markdown implementation for a number of reasons:

- From time to time users who are using the current Markdown formatter ask for a spec / formal list of all supported features. No such thing exists for RedCarpet. There is [CommonMark](https://github.com/ Corporation/CommonMark) but RedCarpet isn't going to support it in the short to medium term (see next point).
- The future development of RedCarpet is uncertain. Few excerpts from a [GitHub issue](https://github.com/ Corporation/RedCarpet), a year ago:

  > **Commonmark won't be supported anytime soon**

  and

  > **A general message about the project for people skimming this thread: I'm sorry Redcarpet isn't really active anymore but my resources are pretty limited since I'm committed to other open source projects and still a student. Feel free to use any existing alternative; this project isn't the great tool it used to be when it was maintained by Vicent.**

- With CommonMark evolving as a Markdown spec that is supported by many implementations and endorsed by organizations like Gitlab and GitHub (which both did the switch from RedCarpet to CommonMarker a while ago), it quickly becomes what users expect when they hear 'Markdown'.
- Migrating existing Textile content is a bit easier since Pandoc has a dedicated Github Flavored Markdown writer module.
- The HTML pipeline approach encourages splitting up the formatting process into it's different aspects (html generation, sanitizing, syntax highlighting etc) which allows for better testability and has potential for future re-use of components with other text formatters. Further, HTML pipeline filters work on actual DOM nodes, making tasks like adding classes to links etc much more straight forward and less prone to bugs than doing so with regular expressions.

Last but not least, this formatter solves a number of currently open issues regarding the RedCarpet based Markdown Formatter:

- #19880 (Incorrect syntax for links in Markdown)
- #20497 (Markdown formatting supporting HTML)
- #20841 (Bare URLs in Markdown don't have "external" class)
- #29172 (Markdown: External links broken)

The main reason why we want to introduce this as a third formatting option and not just replace the existing Markdown formatter is line endings - this formatter does not insert hard breaks (`<br>`) for simple newlines, but requires either a `\` or two spaces at the end of the line to do so. Over time the new formatter could become the default and the RedCarpet based one might be renamed to **Markdown (legacy)** or similar.

English help files are included, and the patch makes sure these are delivered for any other languages as well until corresponding
I really, really like the proposed patch, especially for the HTML Pipeline gem that can help us a lot to improve the current code.

I agree that Redcarpet is not active anymore. The number of commits made in this year is only 3. And I think many Markdown users expect Redmine to behave as CommonMark/GFM compliant because many apps/services that support it.

Let's start discussion to deliver this in 4.2.0.

Love you guys! We are playing with markup format conversions and preparing bulletproof arguments to integrate commonmarker + HTML sanitization for almost half a year. :) Jens did this job perfectly and the only unused argument regarding Redcarpet was #32563. I even tried to make a bulletproof pull request to solve an unpleasant Redcarpet issue observed in our Textile->Markdown migration (since 2013) to test if the project is really dead. It is.

Just a very small improvement is attached.
I'd sign every word written by Jens and would like to share a few remarks in subsequent comments, hope they help.

Go MAEDA, would you mind adding #22323 to related issues regarding the line breaks?

#10 - 2019-12-08 23:46 - Martin Cizek

Temporary workaround before the patch is merged

At the moment, it is possible to use redmine_common_mark plugin. It also allows for configuring commonmarker, but it has no HTML sanitizing implemented. Actually I found this patch when I was about to offer a pull request with html-pipeline to the plugin's author.

We'd still appreciate merging this patch ASAP.

#11 - 2019-12-08 23:47 - Martin Cizek

commonmarker configuration

These are the options used by GitLab. The options in the patch are the same (thumbs up!).

GitHub's configs have a few differences (it does not mean that we want them):
- STRIKETROUGH_DOUBLE_TILDE is not used in repos.
- It uses HARDBREAKS in their issues, which is inconsistent with their wiki and repository rendering.
- They have the tasklist extension enabled. I'd consider enabling it in Redmine, as tasklists is an officially documented GFM feature.

I can imagine that some Redmine users would like other options than us, as we have seen before in Redmine. And it would be quite legit e.g. for the tasklists mentioned above.

Making commonmarker configurable can mean way too many config options, which is difficult to support. A good compromise might be to make a hook for plugins, so that they can change the config.

I can create a follow-up ticket after getting some feedback.

#12 - 2019-12-08 23:48 - Martin Cizek

HTML sanitizing

HTML sanitizing is currently embedded in the commonmark formatter in the patch. That's a good start.

As #807 suggests, sanitizing should rather be a shared concept, eventually with small differences for different formats. GitLab did it this way, their base_sanitization_filter.rb contains common sanitization setup and CommonMark sanitization_filter.rb customizes the HTML::Pipeline::SanitizationFilter on top of that.

I can create a follow-up ticket after getting some feedback.

#13 - 2019-12-08 23:48 - Martin Cizek

Textile to Markdown migration

Pandoc is actually bad at this job. We tried Jens'es redmine_convert_textile_to_markdown (thanks for that!), we ran rendering comparison tests covering hundreds of thousands of strings and the results were poor.

Rendering comparison test = grab all rendered issues and wiki pages from the Textile Redmine instance and a converted-to-Markdown Redmine instance, normalize HTML, compare the HTMLs.

So we forked it, and similarly to Jens, we were adding more and more preprocessors to make Pandoc happy and postprocessors to render it correctly. This is the latest version of the fork. Later, we reworked it completely to a new project, which we'll publish soon.

But if we were doing it again, we would get rid of Pandoc completely. The preprocessing is done by partial rendering using code adapted from Redmine / Redcloth3. The amount of the code is comparable to normal rendering and invoking Pandoc just makes it slow.

Pandoc is a great tool, but this use case is just too specific for a universal format converter.

So the message is just: a good converter exists for Redmine. :)
Hi, thanks for the feedback!

Nice catch about the rendering of Markdown files, that indeed should be switched to commonmark as well. Also great work on the Migrator :) 

General use of HTML Pipeline

It would really make sense to use HTML pipeline for Textile rendering as well, as it will open up a lot of options for refactoring and sharing code (i.e. for HTML sanitization) between the different pipelines via filters. I intentionally stopped before doing this to gauge interest and get commonmark in as quickly as possible. We can (and imho should) rework the Textile rendering at a later point, as well as do some cleanup by moving rendering related code from helpers to the pipeline. The same could be done to the Redcarpet based markdown formatter if we decide to still keep that around at this time.

Configuration

For the sake of not overcomplicating this already complex patch I went with what I think are sensible defaults. I have no idea why Github is so inconsistent with their line break rendering and would definitely stay away from doing the same.

Introducing a way for configuring the rendering pipeline through plugins makes sense, maybe as part of the refactoring when we move over all formatters to the pipeline?

Regarding the task lists - to be honest I left this out since at Planio this would collide with the checklists plugin we have as a standard feature. Also this would require additional code to handle the checking / unchecking of the checkboxes in the rendered text (everywhere markdown is rendered) again making the patch more complex. I feel like this also really only makes sense in issue descriptions and (maybe) wiki pages. So this would be a case where we need slightly different pipeline config depending on the context.

Oh my this patch would be so nice. I know its always hard to say but is there any roadmap when this patch is available in an official release ?

Hans Riekehof wrote in #note-15:

[…] I know its always hard to say but is there any roadmap when this patch is available in an official release ?

Given all the parameters (size/scope/state/complexity of the patch, current Redmine release cycle, current issue scheduling) I'd say not earlier than before the end of 2020. However, this does not mean that it won't be available on the Redmine trunk earlier.

NB: Please keep this issue clean, focused, on-topic and free from superfluous comments...

Thank you for posting the patch.

I think we should remove Redcarpet instead of adding the second Markdown formatter. Although I understand that CommonMaker is not fully compatible with Redcarpet, having two different Markdown formatter introduces some problems.
- It is confusing. Maybe some users cannot understand why Redmine has two Markdown formatter and cannot determine which they should use
- Consumes more memory

Someday after adding support for CommonMaker, we need to remove Redcarpet. In my opinion, when the time we add CommonMaker is the best chance to remove Redcarpet. Replacing Redcarpet with CommonMaker is a simple and understandable story for everyone.

#18 - 2020-02-10 09:18 - Jens Krämer

I agree regarding complexity / confusion of users. So in order to move this forward, should I proceed to change the patch so it directly replaces Redcarpet with CommonMarker for the markdown formatting?

#19 - 2020-02-10 10:53 - Marius BALTEANU

Jens Krämer wrote:

I agree regarding complexity / confusion of users. So in order to move this forward, should I proceed to change the patch so it directly replaces Redcarpet with CommonMarker for the markdown formatting?

I'm my opinion, it is a good move to replace Redcarpet with CommonMarker, but I don't think that we can do it without helping users to migrate their existing content. Gitlab did this kind of migration and they had a very nice process:

https://about.gitlab.com/blog/2019/06/13/how-we-migrated-our-markdown-processing-to-commonmark/

#20 - 2020-02-10 11:43 - Jens Krämer

Well Gitlab is a hosted platform with paying customers, so they had to be extra careful. I don't know if/how that gradual release of commonmark also happened for users of their opensource version on their own servers. From reading that article it appears to me that they did not encounter many issues with existing content since they do not mention having done any automated conversions of Redcarpet data to commonmark.

It would certainly up complexity quite a bit if we were to add support for different formatters at the same time (i.e., rendering old content with Redcarpet and only new content with CommonMarker)...

Gitlab's diff tool could indeed be used as inspiration for a plugin that people could install before upgrading that could give them an idea about what percentage of their contents will render differently at all if they were to upgrade. I wouldn't build an automated converter however - it's really just few corner cases and chances are such a tool breaks more than it fixes.

#21 - 2020-02-21 14:27 - Martin Cizek

- File 0003-CommonMark-external_links_filter.rb.patch added

Attaching an incremental patch to handle invalid URIs in HTML anchors. It prevents HTTP 500 for inputs like:

https://example.com:port/your service

A few remarks:
- This is similar to how GitHub behaves (it keeps href at the link).
- Such links are not marked as external - not sure if it is right or wrong.
- GitLab treats these cases also with removing the link's href, see sanitize_node_link.rb called at base_sanitization_filter.rb#L45 - I consider this a
good approach especially for autolinks, which are still parsed as autolinks, but they are neither blue nor clickable in the end.

**#22 - 2020-02-21 14:43 - Go MAEDA**
- Related to Feature #33037: Support for Gitlab flavored Markdown added

**#23 - 2020-02-21 18:06 - Martin Cizek**

Marius BALTEANU wrote:

> In my opinion, it is a good move to replace Redcarpet with CommonMarker, but I don't think that we can do it without helping users to migrate their existing content.

I believe that keeping the content and just switching the renderer will be the best option for most Redcarpet users. A few reasons for not treating this as a big deal:

- Substantial part of typical rich text contents is quite messy. Users are usually not familiar with the syntax details and they often copy & paste pieces of code, error messages, etc. without enclosing them to code blocks.
- Users often use familiar (i.e. CommonMark) constructs without paying attention to how they exactly render.
- Redmine did breaking changes in text formatting even before. E.g. until 3.4.7, users can write spaces before list items in Textile. In Markdown, superscript syntax was just introduced at some moment and introduction of underline using underscores could have been a surprise too.

I've written [a few more thoughts on this topic here](#).

Formally speaking, conversion will improve things only if the currently rendered output better reflects what was intended to be written than input interpreted as GFM. Which is not the case in my experience. And transforming badly rendered Markdown to GFM would push the rendering problems to the text source.

Disclosure: I have this opinion despite of being the author of a tool which can do the conversion job. :)

**Conversion tool:**
We've published the [redmine_reformat](#) plugin.
It should do a good job for Textile -> Markdown conversion - tested on 250k strings. And it can be also used on common_mark-patched Redmine right away.

The RedcarpetMD -> GFM conversion can be done as MD -> patched Redmine's textilizable() -> HTML -> external Turndown microservice -> GFM (see examples if interested).

Our external Turndown based microservice is still in development and we are hoping for getting close to passing backtranslation tests (MD -> HTML -> MD). Which is quite a complex task and even [bulletproof GFM escaping](#) took almost 1000 lines of Javascript source code.

Having said this, I can only repeat that even a RedcarpetMD -> GFM converter with 100% rendering match would be hardly beneficial to most users.

Sorry for another longer post and hope it helps in decision making. :)

**#24 - 2020-02-23 16:25 - Martin Cizek**

The previous post was a little biased, as we were using patched Redmine's configuration of Redcarpet. The biggest deal in the transition would be the hard_wrap option. Also _underline_ will change meaning to emphasis, and ^superscript^ will not work. I'm still convinced that any eventual conversion would do the best job if it just handles these particular issues at the text source level. We could try to make such converter if it were considered as a dealbreaker.

I'd discourage conversions based purely on pandoc - including the Gitlab's diff tool, which is even not finished according to its source code. :)

**#25 - 2020-02-28 03:23 - Kevin Fischer**
I like this patch a lot, too, and agree that the HTML Pipeline will make existing code much nicer, modular and easier to maintain.

I would also vote for replacing the current markdown parser.

About the compatibility issues:

- **hard_wrap**: Commonmarker has a render option called HARDBREAKS which has the same effect (and this also seems to be the standard behaviour on GitHub)

- **underline** and **superscript** could just be replaces by their direct HTML equivalents (ins and sup) in the text source

---

**#26 - 2020-03-27 10:26 - Jens Krämer**

GitHub uses HARDBREAKS only for rendering markdown in issues and similar (more short form) content. Wiki pages and markdown documents in repositories are rendered without hard line breaks.

Generally not using hard breaks is more sensible if you intend to make your content render properly on a variety of screen widths since it allows paragraphs to flow freely and lines will only end at the end of the available space and not where the original author decided to put a line ending. As such, it is also more important in long form content, less so in shorter snippets of text where it won’t make too much of a difference in terms of readability.

I would prefer to not just continue with forcing hard breaks onto all users of Redmine, but let users choose. It’s important to not only think about existing content here, but also what would be the optimum for new installations.

We could introduce a global setting (i.e. in configuration.yml) where the person setting up a Redmine installation can select either the old behavior (hard breaks everywhere), no hard breaks, or the Github approach with hard breaks in issues and comments but nowhere else. I do not know the exact reasoning that led Github to their decision to have inconsistent hard break rendering across their site, but it might have to do with having content flowing into issues / comments from emails (which is the case in Redmine as well), and/or with avoiding surprises for users (issue creators / commenters) who are unfamiliar with the way markdown line endings work by default.

To handle the issue with incoming emails (which rarely are written with Markdown in mind and therefore often render poorly without hard line breaks turned on) at Planio, we currently enforce hard breaks in content from emails by appending two spaces to each line of text that goes from an email into an issue or comment. This would not be necessary if we had different rendering modes, i.e. hard breaks on in issues / comments, hard breaks off elsewhere. The current architecture however doesn’t really allow for this, the markdown formatter lacks context information. It would be easy to define two different markdown formatting pipelines, one with hard breaks, one without, but still the text formatting view helper has to get some context so further down the chain can be decided which pipeline to use.

---

**#27 - 2020-04-05 12:51 - Marius BALTEANU**

Jens Krämer wrote:

Well Gitlab is a hosted platform with paying customers, so they had to be extra careful. I don’t know if/how that gradual release of commonmark also happened for users of their opensource version on their own servers. From reading that article it appears to me that they did not encounter many issues with existing content since they do not mention having done any automated conversions of Redcarpet data to commonmark.

It would certainly up complexity quite a bit if we were to add support for different formatters at the same time (i.e., rendering old content with RedCarpet and only new content with CommonMarker)...

Gitlab’s diff tool could indeed be used as inspiration for a plugin that people could install before upgrading that could give them an idea about what
percentage of their contents will render differently at all if they were to upgrade. I wouldn’t build an automated converter however - it’s really just few corner cases and chances are such a tool breaks more than it fixes.

Sorry for my late reply on this.

I thought more to an easy way for users to find the content that must be updated, not to fix it automatically. Anyway, in the meantime, I saw Martin Cizek plugin that has some features that can help users with this change. Even if we do not provide any help for users, I’m still in favour of this move because RedCarped project is almost dead and, also, I think we can improve a lot the actual code if we start use the html_pipeline gem.

Jens, what do you think if we try to plug the html_pipeline gem to wiki formatting (both Textile and CommonMark)? I will be very happy to work on moving the current regexes from ApplicationHelper#parse_redmine_links and other methods that currently do content parsing to html pipeline filters. In the end, we would have almost the same pipeline, with only one difference, Textile or CommonMark filter.

Files

<table>
<thead>
<tr>
<th>Patch Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-CommonMark-Markdown-text-formatter.patch</td>
<td>68.5 KB</td>
<td>2019-11-06</td>
<td>Jens Krämer</td>
</tr>
<tr>
<td>0002-attachments_helper-commonmark.patch</td>
<td>725 Bytes</td>
<td>2019-12-08</td>
<td>Martin Cizek</td>
</tr>
<tr>
<td>0003-CommonMark-external_links_filter.rb.patch</td>
<td>808 Bytes</td>
<td>2020-02-21</td>
<td>Martin Cizek</td>
</tr>
</tbody>
</table>