<https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/>

**Markdown Guide**

## Markdown Style Guide for [**about.GitLab.com**](https://about.gitlab.com/)

This website was generated by [Middleman](https://middlemanapp.com/), a blog-aware Static Site Generator ([SSG](https://www.staticgen.com/)) widely used by web developers.[Markup language](https://en.wikipedia.org/wiki/Markup_language) is part of the structure of any SSG. It is a system to write documents making them somehow syntactically distinguishable from text. [Lightweight markup languages](https://en.wikipedia.org/wiki/Lightweight_markup_language) have a simplified and unobtrusive syntax, designed to be easily written within any text editor. That's what we use to write our content. The majority of SSGs use markdown engines for this purpose. Read through our blog post on [Modern Static Site Generators](https://about.gitlab.com/2016/06/10/ssg-overview-gitlab-pages-part-2/) to understand how they work.

For [about.GitLab.com](https://about.gitlab.com/) we use [kramdown](http://kramdown.gettalong.org/), which is an advanced Markdown engine with a lot of interesting features that most of the other engines don't have, such as [inline attribute lists](https://kramdown.gettalong.org/syntax.html#inline-attribute-lists) (IALs), which enable easy styling beyond the standard Markdown options.

If you never have written a single line in markdown markup, don't worry, it's easy to learn and even easier to use. You'll probably be surprised how handy it is once you get used to it. And you'll miss it whenever the tech you're using doesn't support markdown.

In most of GitLab text areas you'll find markdown support. Not all of them run with kramdown, so the markup will not behave equally "GitLabwide". For **GitLab.com**, **GitLab CE** and **GitLab EE** text areas, the markdown engine is currently[CommonMarker](https://github.com/gjtorikian/commonmarker). Here you can find the [markdown style guide](https://gitlab.com/help/user/markdown.md) for them.

This guide has been made to make it easier for everyone to use kramdown features and save a lot of time writing content for [about.GitLab.com](https://about.gitlab.com/), including handbook pages, website pages, blog posts and everything else within the project [www-GitLab-com](https://gitlab.com/gitlab-com/www-gitlab-com).

There are different possible syntaxes for most of the markups described below, but this guide is to be considered the standard for [about.GitLab.com](https://about.gitlab.com/).

***Note:****this document is maintained by the*[*Technical Writing*](https://about.gitlab.com/handbook/product/technical-writing/)*Team.*

### Blog

For our [Blog](https://about.gitlab.com/blog/), everything in this guide can be applied. Read through the [Blog Style Guidelines](https://about.gitlab.com/handbook/marketing/blog/#styles-guidelines) for further information.

## Headings

## Heading h2

### Heading h3

#### Heading h4

**Output**

## Heading h2

### Heading h3

#### Heading h4

Notes:

* We don't use h1 headings, as they already are displayed on every page as its title, and we should not apply more than one h1 per page.

When you use a top level heading, or an <h1>, you’re setting up a semantic relationship between that heading and the remainder of the content on a page, describing what it is about. If you then use a second <h1> on the same page, you’re creating some potential confusion, because someone, or a search engine might see that as the ending of the semantic relationship between the content after the first <h1> and the start of this new <h1>. [SEO Guide](http://www.seobythesea.com/2012/01/heading-elements-and-the-folly-of-seo-expert-ranking-lists/)

* Always start with h2 (##), and respect the order h2 → h3 → h4. Never skip the hierarchy level, such as h2 → h4.

The six heading elements, H1 through H6, denote section headings. Although the order and occurrence of headings is not constrained by the HTML DTD, documents ***should not skip levels*** (for example, from H1 to H3), as converting such documents to other representations is often problematic. [W3C](https://www.w3.org/MarkUp/html-spec/html-spec_5.html#SEC5.4)

* Always leave a blank space between the hash # and the text next to it, otherwise it won't render properly.
* For keeping the text clear and the markdown consistent, [jump a line](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#jump-a-line) between any heading and its subsequent paragraph.

## Paragraphs, breaks, and horizontal lines

Regular paragraphs are obtained by just writing text lines. If you hit **enter** between two lines, both lines will be joined into a single paragraph, which is called [wrapping text](https://about.gitlab.com/2016/10/11/wrapping-text/#do-wrap-it). But, if you leave a blank line between them, they will split into two paragraphs.

It is a best practice to write each a paragraph on a single line. Don't put line breaks at the end of each sentence, and don't break up lines at a certain character limit. It is very difficult to review and edit copy with artificial lines breaks.

In some Git tools, diffs in future MRs may be easier to understand with additional line breaks, however GitLab's web interface as well as many desktop Git tools feature substring change highlighting within lines and side-by-side or similar version comparison so there is no need for artificial line breaks.

### Wrapping Text

We usually break the lines within paragraphs to facilitate reviews. Do not leave blank spaces after the last word of the line broken within a paragraph, unless you want it to be intentionally broken with a <br>.

#### Regular paragraphs and automatic join

This text is a paragraph.

This won't be another paragraph, it will join the line above it.

This will be another paragraph, as it has a blank line above it.

**Output**

This text is a paragraph. This won't be another paragraph, it will join the line above it.

This will be another paragraph, as it has a blank line above it.

### Additional breaks

In case you need an additional break (or some extra space between lines), you can simply use the HTML break tag <br>, leaving blank lines above and below it:

Text A

*<!-- blank line -->*

<br>

*<!-- blank line -->*

Text B

**Output**

Text A

Text B

### Horizontal lines

A sequence of three or more dashes will produce a horizontal line, but let's use always **4** as standard. Leave blank lines after and before it:

Text

*<!-- blank line -->*

----

*<!-- blank line -->*

Text

**Output**

Text

Text

## Emphasis: bold and italic

To display bold or italic text, wrap it in stars or underlines:

This is **\*\*bold\*\*** and this is \_italic\_.

**Output**

This is **bold** and this is italic.

## Links

There are a few different ways to display links with markdown markup, but to keep some standards, let's try to use the following options only.

Important notes:

* Don't take it as a restrictive rule, but [avoid using meaningless text for links](https://about.gitlab.com/handbook/communication/#writing-style-guidelines) as "this article" or "read here." The link text should be meaningful even if taken out of context; this makes the links more useful and accessible for people using screen readers.
* Check for broken links: <http://www.deadlinkchecker.com/>

### Inline Links

We'd rather use inline links, such as [Text to display](link), as they are easier to maintain.

### Relative links

Use relative links when referring to links found on [about.gitlab.com](https://about.gitlab.com/). For example, a link to our blog handbook should look like this /handbook/marketing/blog/ and **not** this https://about.gitlab.com/handbook/marketing/blog/. Remove everything from the https through about.gitlab.com, but retain the forward slash after .com.

For links to GitLab.com or anywhere else you must use the entire link, including the http:.

### Mailto links

If you're adding an email address to a page be sure to format your link with mailto to avoid creating broken links. For example, [example@gitlab.com](mailto:example@gitlab.com)

### Identifiers

When there are **repeated** links across a single page, you can opt for using identifiers.

Place the identifiers at the end of the paragraph (or the section), arranging them in alphabetical order.

[Text to display][identifier] will display a link.

[Another text][another-identifier] will do the same. Hover the mouse over it to see the title.

[This link] will do the same as well. It works as the identifier itself.

[This link][] (same as above), has a second pair of empty brakets to indicate that the following parenthesis does not contain a link.

<https://example.com> works too. Must be used for explicit links.

*<!-- Identifiers, in alphabetical order -->*

[another-identifier]: https://example.com "This example has a title"

[identifier]: http://example1.com

[this link]: http://example2.com

**Output**

[Text to display](http://example1.com/) will display a link.

[Another text](https://example.com/) will do the same. Hover the mouse over it to see the title.

[This link](http://example2.com/) will do the same as well. It works as the identifier itself.

[This link](http://example2.com/) (same as above), has a second pair of empty brackets to indicate that the following parenthesis does not contain a link.

[https://example.com](https://example.com/) works too. Must be used for explicit links.

Note:

* Identifiers **are not** case sensitive. They can be single words as [link] or [multiple words too].

## Lists

Both ordered and unordered lists are very straight-forward to produce. There are a few ways to produce the same results, but let's stick with the following, again, to maintain some standards.

Always use **3** blank spaces to indent a nested list (to create sub items).

Tip: don't leave blank lines **between the items**, unless you have a reason to do so.

**Important:** always leave a blank line between the paragraph or heading and the subsequent list! If you don't, the list will not render.

### Ordered lists

Ordered lists are pretty easy to create. Couldn't be more intuitive:

Paragraph:

1. Item one

1. Sub item one

2. Sub item two

3. Sub item three

2. Item two

**Output**

Paragraph:

1. Item one
   1. Sub item one
   2. Sub item two
   3. Sub item three
2. Item two

To be practical and avoid errors on the numbers, use "1" for all the items. The markdown engine will output them in the correct order.

Paragraph:

1. Item one

1. Sub item one

1. Sub item two

1. Item two

1. Item three

**Output**

Paragraph:

1. Item one
   1. Sub item one
   2. Sub item two
2. Item two
3. Item three

### Unordered lists

Unordered lists are very easy to create too:

Paragraph:

- Item 1

- Item 2

- Item 3

- Sub item 1

- Sub item 2

- Item 4

**Output**

Paragraph:

* Item 1
* Item 2
* Item 3
  + Sub item 1
  + Sub item 2
* Item 4

### Split lists

Let's say, for some reason, you want to split a list in different parts. To do that, use the markup ^ to indicate the end of a list and the beginning of the next:

- list one - item 1

- list one - item 2

- sub item 1

- sub item 2

- list one - item 3

^

- list two - item A

- list two - item B

^

- list three - item \_i\_

- list three - item \_ii\_

**Output**

* list one - item 1
* list one - item 2
  + sub item 1
  + sub item 2
* list one - item 3
* list two - item A
* list two - item B
* list three - item i
* list three - item ii

## Images

To insert images to your markdown file, use the markup ![ALT](/path/image.ext). The path can either be relative to the website, or a full URL for an external image. The supported formats are .png, .jpg, .gif. You might be able to use some .svg files too, depending on its structure.

![Semantic description of image](/images/path/to/folder/image.png "Image Title")

You can also use an identifier, as we do for [links](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#links):

![Semantic description of image][identifier]

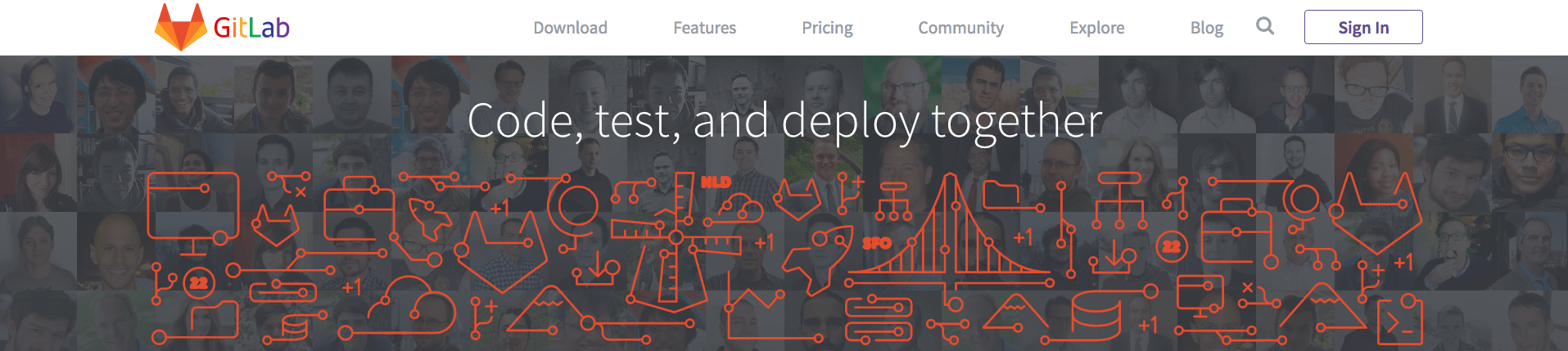
If you want to add a caption to your image, it's easily achieved with:

![Semantic description of image](/images/path/to/folder/image.png)*\*My caption\**

For clickable images, simply wrap the image markup into a [link markup](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#links):

[![Semantic description of image](/images/path/to/folder/image.png "Hello World")*\*My caption\**][about.gitlab.com]

**Output**

[[](https://about.gitlab.com/)My caption](https://about.gitlab.com/)

**Important notes:**

* Apply [shadow](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#shadow) to your images!
* All images must be placed [under /source/images/](https://gitlab.com/gitlab-com/www-gitlab-com/tree/master/source/images), in an appropriate directory. Only screenshots and public domain images are permitted.
* The text inside the square brackets is an image attribute called ALT, which stands for alternative text. [Including descriptive alt text](https://webaim.org/techniques/alttext/) helps maintain accessibility for every visitor and should always be included with an image. When you add alt text be sure to describe the content and function of an image. In addition to the accessibility benefits, ALT is useful for SEO, and it is displayed when, for some reason, that image is not loaded by the browser.
* For the same reasons, the image must contain a name related to it. Example: instead of image-01.jpg, name it black-dog.jpg, if it's a photo of a black dog.
* It's also recommendable adding an image title, as the "Hello World" exemplified above.

## Diagrams

There are two ways to insert diagrams via Markdown:

1. Mermaid
2. PlantUML

### Mermaid

See the [examples in the GitLab docs](https://docs.gitlab.com/ee/user/markdown.html#diagrams-and-flowcharts-using-mermaid) on how to use Mermaid.

### PlantUML

You can use [PlantUML](http://plantuml.com/) in Markdown blocks. For example:

```plantuml

!define ICONURL https://raw.githubusercontent.com/tupadr3/plantuml-icon-font-sprites/v2.1.0

skinparam defaultTextAlignment center

!include ICONURL/common.puml

!include ICONURL/font-awesome-5/gitlab.puml

!include ICONURL/font-awesome-5/java.puml

!include ICONURL/font-awesome-5/rocket.puml

!include ICONURL/font-awesome/newspaper\_o.puml

FA\_NEWSPAPER\_O(news,good news!,node) #White {

FA5\_GITLAB(gitlab,GitLab.com,node) #White

FA5\_JAVA(java,PlantUML,node) #White

FA5\_ROCKET(rocket,Integrated,node) #White

}

gitlab ..> java

java ..> rocket

```

**Output**

## Videos

There are two ways of displaying videos: within HTML5 <video> tags and within <iframe> tags.

### Display videos from YouTube

This method works for YouTube videos and any other embed video within an <iframe> tag.

1. Copy the code below and paste it into your markdown file. Leave a blank line above and below it. **Do NOT edit the code block** (e.g., remove spaces - the video iframe may not render properly)
2. Go the video URL you want to display
3. Click on "Share", then "Embed"
4. Copy the <iframe> source (src) URL **only**, and paste it replacing the src below:

*<!-- blank line -->*

<figure class="video\_container">

<iframe src="https://www.youtube.com/embed/enMumwvLAug" frameborder="0" allowfullscreen="true"> </iframe>

</figure>

*<!-- blank line -->*

**Output**

*Trouble viewing this resource? You may need to update your*[*cookie settings*](javascript:Cookiebot.renew();)*to allow****personalization****cookies.*

### Display local videos (HTML5)

This method works for any video uploaded to somewhere retrievable from the internet from a URL, or from a relative path like path/to/video.mp4.

1. Read through the [w3schools HTML5 video guide](http://www.w3schools.com/tags/tag_video.asp), or the [MDN <video> guide](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/video).
2. Record or export the video in these three formats to achieve cross-browser and cross-device compatibility: .mp4, .ogg and .webm.
3. Get the URL for your video
4. Choose an image to use as a poster
5. Copy the code below and paste it to your file
6. Replace the src URLs for your video URLs

*<!-- blank line -->*

<figure class="video\_container">

<video controls="true" allowfullscreen="true" poster="path/to/poster\_image.png">

<source src="path/to/video.mp4" type="video/mp4">

<source src="path/to/video.ogg" type="video/ogg">

<source src="path/to/video.webm" type="video/webm">

</video>

</figure>

*<!-- blank line -->*

**Output**

***Note:*** in case you don't have all formats recommended by ***w3schools***, you can use just one of them, but your video most likely won't be supported in all devices and browsers. The video above (*.mp4* only) worked on Mozilla Firefox for OS X, Android, and Windows, and on Chrome for Android and for Windows. But it may not work on other devices/browser, such as Chrome for OS X and iOS, or Safari. In fact, the best option is using YouTube or Vimeo embed videos in *<iframe>* tags.

### Display other videos

For any other videos, such as from Vimeo or Google Drive, grab the video iframe only, and proceed like we do for [YouTube videos](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#display-videos-from-youtube), wrapping the <iframe> within a <figure class="video\_container">, for responsiveness. Copy and paste the code below, replacing **only** the iframe URL with your own:

*<!-- blank line -->*

<figure class="video\_container">

<iframe src="https://drive.google.com/file/d/0B6m34D8cFdpMZndKTlBRU0tmczg/preview" frameborder="0" allowfullscreen="true"> </iframe>

</figure>

*<!-- blank line -->*

**Output**

### Display multiple videos

To display multiple videos on the same page, just repeat the figure code block where you want them to show up, replacing the video ID with the respective ID corresponding to your videos.

To display multiple videos in a sequence, just copy the figure code block and paste it as many times as necessary. Always leave a blank line between the blocks. Do **NOT** remove the spaces, otherwise your videos may not render properly.

*<!-- blank line -->*

<figure class="video\_container">

<iframe src="https://drive.google.com/file/d/0B6m34D8cFdpMZndKTlBRU0tmczg/preview" frameborder="0" allowfullscreen="true"> </iframe>

</figure>

<figure class="video\_container">

<iframe src="https://drive.google.com/file/d/0B6m34D8cFdpMZndKTlBRU0tmczg/preview" frameborder="0" allowfullscreen="true"> </iframe>

</figure>

<figure class="video\_container">

<iframe src="https://drive.google.com/file/d/0B6m34D8cFdpMZndKTlBRU0tmczg/preview" frameborder="0" allowfullscreen="true"> </iframe>

</figure>

*<!-- blank line -->*

## Table of Contents (ToC)

With kramdown, creating a Table of Contents is the easiest thing ever! The automatic ToC will include every heading in the document, unless you don't want it to be included. You do not need to add anchors individually to every title. This is an automated process.

----

## On this page

{:.no\_toc}

- TOC

{:toc}

----

As always, leave a blank line before and after the markup. Note that there are four dashes beginning and closing the block, which is not required, but recommendable for keeping the same standards throughout [about.GitLab.com](https://about.gitlab.com/).

The heading "On this page" can be adapted to your case, e.g., "In this tutorial", or "In this guide", etc. It's not required either, but recommended.

The markup {:.no\_toc} is used every time you don't want to include a heading into the ToC. Just add it right below the heading, and it won't be included into the ToC. In fact no\_toc is a [custom class](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#classes-ids-and-attributes), as described later in this guide.

The **output** ToC can be found at the very beginning of this page.

Alternatively, you can use ordered ToC too, displaying numbers at the beginning of the list. Just use the markup for ordered lists and kramdown will be smart enough to understand what you want:

1. TOC

{:toc}

## Tables

Tables for markdown are challenging. So, we have two possible approaches: use markdown whenever possible, but if you need pretty advanced table layouts, you are free to add them in HTML markup instead.

Markdown is not a replacement for HTML, or even close to it. ([*John Gruber*](http://daringfireball.net/projects/markdown/syntax#html))

As explained by John Gruber, the creator of markdown, it was not created to replace HTML, so there are situations we can't avoid using HTML. With complex tables, that's the case.

The following table has a header (first line), then markup to define the desired alignment (dashes and colons), then the table body. You can go ahead and add separators to create subsequent table bodies.

However you prepare your table, its design will depend upon the CSS styles defined for them.

The last markup {: .custom-class #custom-id} **can** be used in case you want to attribute a [custom class and/or a custom ID](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#classes-ids-and-attributes) to the <table> element.

| Default aligned | Left aligned | Center aligned | Right aligned |

|-----------------|:-------------|:---------------:|---------------:|

| First body part | Second cell | Third cell | fourth cell |

| Second line | foo | **\*\*strong\*\*** | baz |

| Third line | quux | baz | bar |

|-----------------+--------------+-----------------+----------------|

| Second body | | | |

| 2nd line | | | |

|-----------------+--------------+-----------------+----------------|

| Third body | | | Foo |

{: .custom-class #custom-id}

**Output**

| **Default aligned** | **Left aligned** | **Center aligned** | **Right aligned** |
| --- | --- | --- | --- |
| First body part | Second cell | Third cell | fourth cell |
| Second line | foo | **strong** | baz |
| Third line | quux | baz | bar |
| Second body |  |  |  |
| 2nd line |  | Hello World |  |
| Third body |  |  | Foo |

Certain tools can help you to create your own complex table if you need merging lines or columns, and more advanced layouts. This is a [Table Generator](http://www.tablesgenerator.com/html_tables) that might help you out.

Note that the bars, spaces, and dashes were used symmetrically above just for providing a nice view of the table markup. The symmetry is not required.

Read through the [kramdown syntax guide](http://kramdown.gettalong.org/syntax.html#tables) on tables for further information.

## Code blocks

There are a few options for displaying code blocks with kramdown. Most of them use backticks `.

### In-line

This is an `in-line` code block.

**Output**

In-line

This is an in-line code block.

### Fenced

```

def hello

puts "Hello world!"

end

```

Fenced Highlighted

```ruby

def hello

puts "Hello world!"

end

```

or

```

def hello

puts "Hello world!"

end

```

{: .language-ruby}

**Output**

Fenced

def hello

puts "Hello world!"

end

Fenced Highlighted

**def** **hello**

puts "Hello world!"

**end**

or

**def** **hello**

puts "Hello world!"

**end**

### Indented

Add 4 white spaces before every line:

def hello

puts "Hello world!"

end

Indented Highlighted

def hello

puts "Hello world!"

end

{: .language-ruby}

**Output**

Indented

def hello

puts "Hello world!"

end

Indented Highlighted

**def** **hello**

puts "Hello world!"

**end**

### Nested

Add 4 white spaces before every line. This is used when you want to display a code block within a code block.

```

def hello

puts "Hello world!"

end

```

**Output**

Nested

```

def hello

puts "Hello world!"

end

```

### In lists

Indent the text of each item in 3 white spaces. Leave blank lines between the code block and the list items, and ident the code block in 5 white spaces:

1. Item 1

1. Item 2

```ruby

def hello

puts "Hello world!"

end

```

1. Item 3

**Output**

1. Item 1
2. Item 2
3. **def** **hello**
4. puts "Hello world!"
5. **end**
6. Item 3

## Blockquotes

Blockquotes are good resources to mentioning someone else's quotes, like we did [just above](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#quote). Also, can be used to emphasize a sentence or a small paragraph.

> This is a blockquote.

> On multiple lines.

That may be lazy.

>

> This is the second paragraph.

----

> This is a paragraph.

>

> > A nested blockquote.

>

> ### Headers work

>

> \* lists too

>

> and all other block-level \*\*elements\*\*.

>

> Even code blocks:

>

> def hello

> puts "Hello world!"

> end

> {: .language-ruby}

**Output**

This is a blockquote. On multiple lines. That may be lazy.

This is the second paragraph.

This is a paragraph.

A nested blockquote.

### Headers work

* lists too

and all other block-level **elements**.

Even a code block:

**def** **hello**

puts "Hello world!"

**end**

## Notes

This is a regular paragraph.

**\*\*Note:\*\*** a note is something that needs to be mentioned but is apart from the context.

{: .note}

**Output**

This is a regular paragraph.

***Note:****a note is something that needs to be mentioned but is apart from the context.*

## Comments

Markdown markup

This is a paragraph

{::comment}

This is a comment which is

completely ignored.

{:/comment}

... paragraph continues here.

Regular HTML markup

*<!-- This is accepted as a comment too -->*

**Output**

This is a paragraph … paragraph continues here.

## Anchors

Add an anchor anywhere with:

[](){: name="hello-world"}

Some text

Or simply use an ID:

Some text

{: #hello-world}

## Font Awesome

Yes, we can use fancy [Font Awesome](http://fontawesome.io/icons/) icons too.

Regular

### <i class="fas fa-puzzle-piece" aria-hidden="true"></i> Puzzle Icon

{: #puzzle}

And you can go further, such as the following.

Styled

### <i class="fab fa-gitlab fa-fw" style="color:rgb(107,79,187); font-size:.85em" aria-hidden="true"></i> Purple GitLab Tanuki

{: #tanuki-purple}

### <i class="fab fa-gitlab fa-fw" style="color:rgb(252,109,38); font-size:.85em" aria-hidden="true"></i> Orange GitLab Tanuki

{: #tanuki-orange}

**Output**

Regular

### Puzzle Icon

Styled

### Purple GitLab Tanuki

### Orange GitLab Tanuki

When doing something like this to a heading, it's important give it a custom ID (e.g., {: #puzzle}), otherwise the one automatically created by kramdown will sound very awkward.

The class fa-fw is used when you want to display the icons as a list. They will be aligned, as well as the text right beside them.

See live examples [on this post](https://about.gitlab.com/2016/06/10/ssg-overview-gitlab-pages-part-2/), where the icons are used to illustrate the text.

## Classes, IDs, and attributes

Defining CSS classes, and elements IDs and attributes with markdown is definitely something unusual (kramdown magic!).

But yes, if you know how to use it, you'll love it! Check how easy it is to do that with kramdown:

Paragraph

{: .class .class-1 .class-2}

Paragraph

{: #custom-id}

Paragraph

{: .class .class-1 #custom-id-1}

## Heading

{: .class .class-1 #custom-id-2}

Paragraph

{: .class #custom-id-3 style="padding-top:0" key="value"}

## Heading {#hello}

List:

- {: .class} List item with custom class

- {:#id} List item with custom id

To a [link]{: #link}, in-line.

This is a [link][google-es]{:hreflang="es"} in Spanish.

[link]: https://google.com

[google-es]: https://google.es

**Output**

Paragraph

Paragraph

Paragraph

### Heading

Paragraph

### Heading

List:

* List item with custom class
* List item with custom id

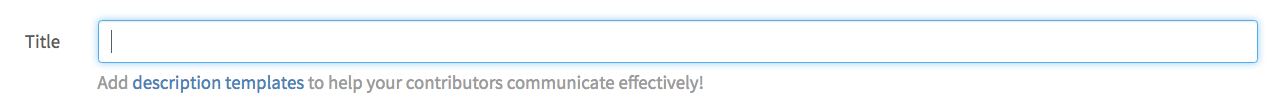
To a [link](https://google.com/), in-line.

This is a [link](https://google.es/) in Spanish.

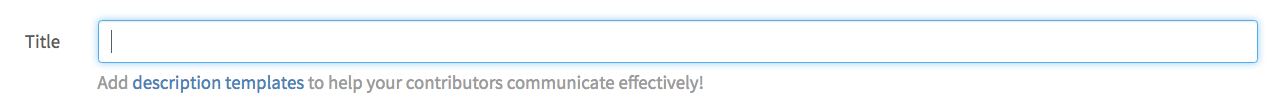
### Special classes

#### Shadow

The CSS class called shadow should be used when your image edges are not clearly defined. This happens when it has a white background or when it's a screenshot with text (for example, a screenshot of our user interface). For example, this image can be mistaken as part of the text:



Now, if you apply the class shadow to the image, it's discreetly highlighted from the text:



To do that, apply the class directly to the image by adding the markup {: .shadow} right after the image markup:

![image alternative text](/path/to/image.png){: .shadow}

#### Note

As [previously](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#notes) explained, you can add the class note to paragraphs that you don't want to call attention to:

***Note:****this is something I don't want to call attention to.*

Markup:

**\*\*Note:\*\*** this is something I don't want to call attention to.

{: .note}

#### Colors

Add a custom class to a heading or paragraph using the following special classes.

**GitLab Orange**

#### GitLab Orange Heading

Markup:

### GitLab Orange Heading

{: .gitlab-orange}

**GitLab Purple**

#### GitLab Purple Heading

Markup:

### GitLab Purple Heading

{: .gitlab-purple}

#### Text Align

By default, the text will always be aligned to the left. You have a few options to customize it with custom classes:

* Center: .text-center
* Right: .text-right
* Justify: .text-justify

For demonstrations purposes, they are aligned in an [alert box](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#colorful-sections), but this can be applied to regular paragraphs and headings.

**Align to the center**

Center-aligned

Alert box markup:

Center-aligned

{: .alert .alert-info .text-center}

Paragraph markup:

Center-aligned

{: .text-center}

Heading markup:

### Center-aligned

{: .text-center}

**Align to the right**

Right-aligned

Right-aligned

{: .alert .alert-info .text-right}

**Justify**

Justified

Justified

{: .alert .alert-info .text-justify}

## Mix HTML + Markdown Markup

Mixing HTML markup with markdown is something almost "unthinkable" to someone used to regular markdown. And it's all over this document!

Use the following markup at the beginning of your document:

{::options parse\_block\_html="true" /}

And feel free to mix everything up:

Something in \*\*markdown\*\*.

<p>Then an HTML tag with crazy \*\*markup\*\* \_all over\_ the place!</p>

**Output**

Something in **markdown**.

Then an HTML tag with crazy **markup** all over the place!

You can close the markup parser tag at any point, if you want to:

{::options parse\_block\_html="false" /}

## Colorful sections

To add notes and warning blocks into colorful boxes, we are making use of Bootstrap's [panel blocks](https://getbootstrap.com/components/#panels-alternatives) and [alert boxes](https://getbootstrap.com/components/#alerts).

Colorful sections are applied for very specific purposes and must not be overused.

Use panels when your description contains more than one paragraph, or a long paragraph. For single and short paragraphs, use alert boxes instead.

When using panels, make sure to add the HTML parser markup to the beginning of your document's body: {::options parse\_block\_html="true" /}.

Copy paste the following code according to what you want to present to the user and replace only the description. The available colors are: blue (info), green (success), amber (warning) and red (danger), as follows.

### Additional Information

Use the following code for **important notes** and additional information:

<div class="panel panel-info">

\*\*Note\*\*

{: .panel-heading}

<div class="panel-body">

NOTE DESCRIPTION

</div>

</div>

To apply to a single paragraph, use an alert box:

My important paragraph.

{: .alert .alert-info}

Blue panels render like:

**Note**

NOTE DESCRIPTION

And blue alert boxes render like:

My important paragraph.

If you want the text inside the alert box to be blue as well, we need to apply [custom styles](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#styles) to the markdown document. They will override the existing ones. Add the following style tag to the end of your file.

**<**style**>**

**.alert-info** {

**color**: rgb(49,112,143) **!important**;

}

**</**style**>**

### Warnings

Use the following code for warnings, like information that may have a different result if not used correctly:

<div class="panel panel-warning">

\*\*Warning\*\*

{: .panel-heading}

<div class="panel-body">

WARNING DESCRIPTION

</div>

</div>

To apply to a single paragraph, use an alert box:

My warning paragraph.

{: .alert .alert-warning}

Amber panels render like:

**Warning**

WARNING DESCRIPTION

And amber alert boxes render like:

My warning paragraph.

If you want the text inside the alert box to be amber as well, we need to apply [custom styles](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#styles) to the markdown document. They will override the existing ones. Add the following style tag to the end of your file.

**<**style**>**

**.alert-warning** {

**color**: rgb(138,109,59) **!important**;

}

**</**style**>**

### Danger

Use the following code for crucial warnings, like commands that result in loss of data:

<div class="panel panel-danger">

\*\*Danger\*\*

{: .panel-heading}

<div class="panel-body">

DANGER DESCRIPTION

</div>

</div>

To apply to a single paragraph, use an alert box:

My danger paragraph.

{: .alert .alert-danger}

Red panels render like:

**Danger**

DANGER DESCRIPTION

And red alert boxes render like:

My danger paragraph.

If you want the text inside the alert box to be red as well, we need to apply [custom styles](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#styles) to the markdown document. They will override the existing ones. Add the following style tag to the end of your file.

**<**style**>**

**.alert-danger** {

**color**: rgb(169,68,66) **!important**;

}

**</**style**>**

### Do's and Don'ts

You can use the combination of green and red panels or alert boxes for "Do's and Don'ts":

<div class="panel panel-success">

\*\*Do's\*\*

{: .panel-heading}

<div class="panel-body">

THINGS TO DO

</div>

</div>

or, use an alert box:

TO DO.

{: .alert .alert-success}

Not to do:

<div class="panel panel-danger">

\*\*Don'ts\*\*

{: .panel-heading}

<div class="panel-body">

THINGS NOT TO DO

</div>

</div>

or, use an alert box:

NOT TO DO.

{: .alert .alert-danger}

By doing so, the green panels for "DO'S" will look like:

**Do's**

THINGS TO DO

or, if you chose an alert box:

TO DO.

If you want the text inside the alert box to be green as well, we need to apply [custom styles](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#styles) to the markdown document. They will override the existing ones. Add the following style tag to the end of your file.

**<**style**>**

**.alert-green** {

**color**: rgb(60,118,61) **!important**;

}

**</**style**>**

And for your "DON'TS" within red panels will look like:

**Don'ts**

THINGS NOT TO DO

or, if you chose a red alert box:

NOT TO DO.

### Custom alert panels and alert boxes

All the previously mentioned alert boxes and panels are available by default by [Bootstrap](http://getbootstrap.com/components/#alerts). If we want them in a different color, we need [custom styles](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#styles). At [about.GitLab.com](https://about.gitlab.com/), we can use the orange and the purple one, as follows.

When using panels, don't forget to add to the beginning of your file the [HTML parser markup](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#html-parser) to be able to mix HMTL + Markdown: {::options parse\_block\_html="true" /}.

#### GitLab Orange Alert Panel

**Heading**

Text in markdown.

Panel block markup:

<div class="panel panel-gitlab-orange">

\*\*Heading\*\*

{: .panel-heading}

<div class="panel-body">

Text in markdown.

</div>

</div>

#### GitLab Orange Alert Box

My text in an orange box.

Box block markup:

My text in an orange box.

{: .alert .alert-gitlab-orange}

#### GitLab Purple Alert Panel

**Heading**

Text in markdown.

Panel block markup:

<div class="panel panel-gitlab-purple">

\*\*Heading\*\*

{: .panel-heading}

<div class="panel-body">

Text in markdown.

</div>

</div>

#### GitLab Purple Alert Box

My text in an purple box.

Box block markup:

My text in an purple box.

{: .alert .alert-gitlab-purple}

### GitLab Webcast Alert Box

To be used in a CTA for webcast announcement in blog posts. You can use it for other purposes as well. Use it together with the [HMTL parser](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#html-parser):

 The webcast I want to announce - [Register here](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/)!

{::options parse\_block\_html="true" /}

<i class="fab fa-gitlab" style="color:rgb(107,79,187); font-size:.85em" aria-hidden="true"></i>&nbsp;&nbsp;

The webcast I want to announce - [Register here][webcast-link]!

&nbsp;&nbsp;<i class="fab fa-gitlab" style="color:rgb(107,79,187); font-size:.85em" aria-hidden="true"></i>

{: .alert .alert-webcast}

## Styles

Yes, guess what?

This:

**<**style**>**

**.purple** {

**color**:inherit;

}

**.purple:hover** {

**color**:rgb(107,79,187);

}

**</**style**>**

Plus:

Hey! Hover the cursor over me and guess what?! :)

{: .purple}

Equals to:

**Output**

Hey! Hover the cursor over me and guess what?! :)

And yes, the <style> tag is in this very markdown file. Believe it or not!

## Embed documents

It's easy to embed Google Docs, Sheets, Slides, and pretty much everything that provides an iframe to use with. The only thing you need to do is use the following code inside your markdown file and replace the iframe from the document you want to embed:

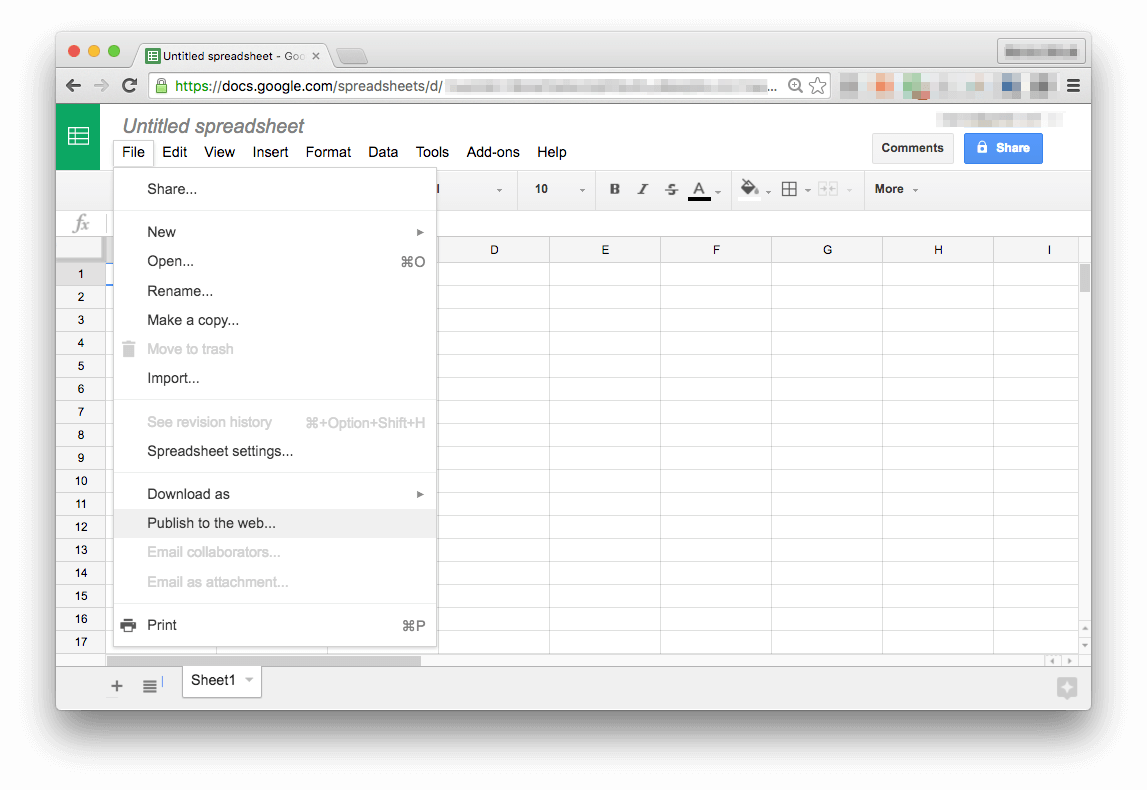
<figure class="video\_container">

<iframe IFRAME CONTENT></iframe>

</figure>

### Google products

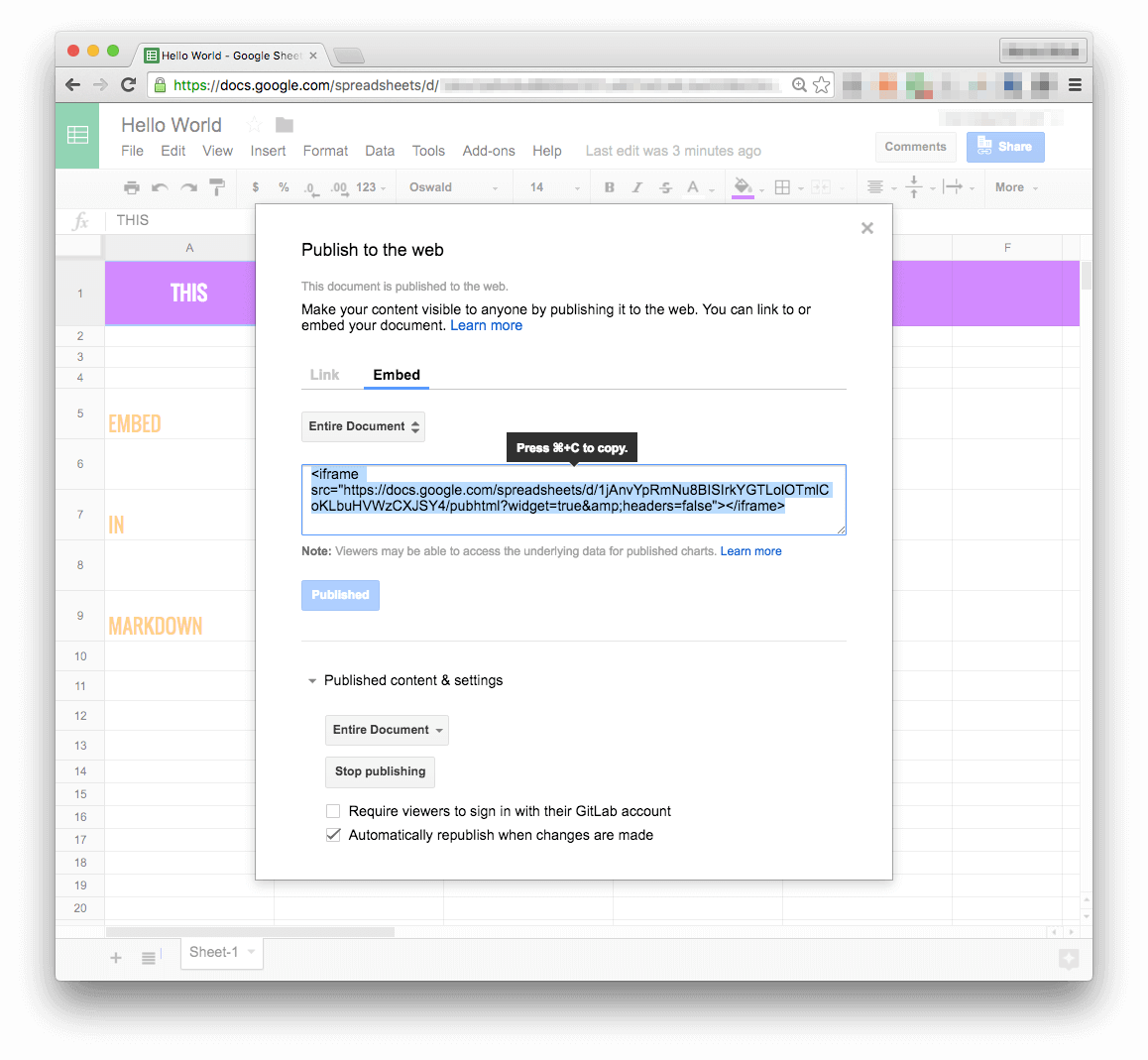
For Google products, with your document opened, click **File** -> **Publish to the web**. For example here's how Google sheets will look like:



Choose **Embed**, check your settings, click on **Publish** and copy the <iframe>. Then go to your markdown file and wrap the iframe into a <figure> tag with the responsive video\_container class, as shown [in the beginning](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#embed-documents).

#### Google Sheets

Let's exemplify with this [simple spreadsheet](https://docs.google.com/a/gitlab.com/spreadsheets/d/1jAnvYpRmNu8BISIrkYGTLolOTmlCoKLbuHVWzCXJSY4/edit?usp=sharing). Follow the info [above](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#google-products) to find the iframe:



Copy the code below and paste to your markdown file (leave a blank line above and below it). Then replace the <iframe>with your own:

<figure class="video\_container">

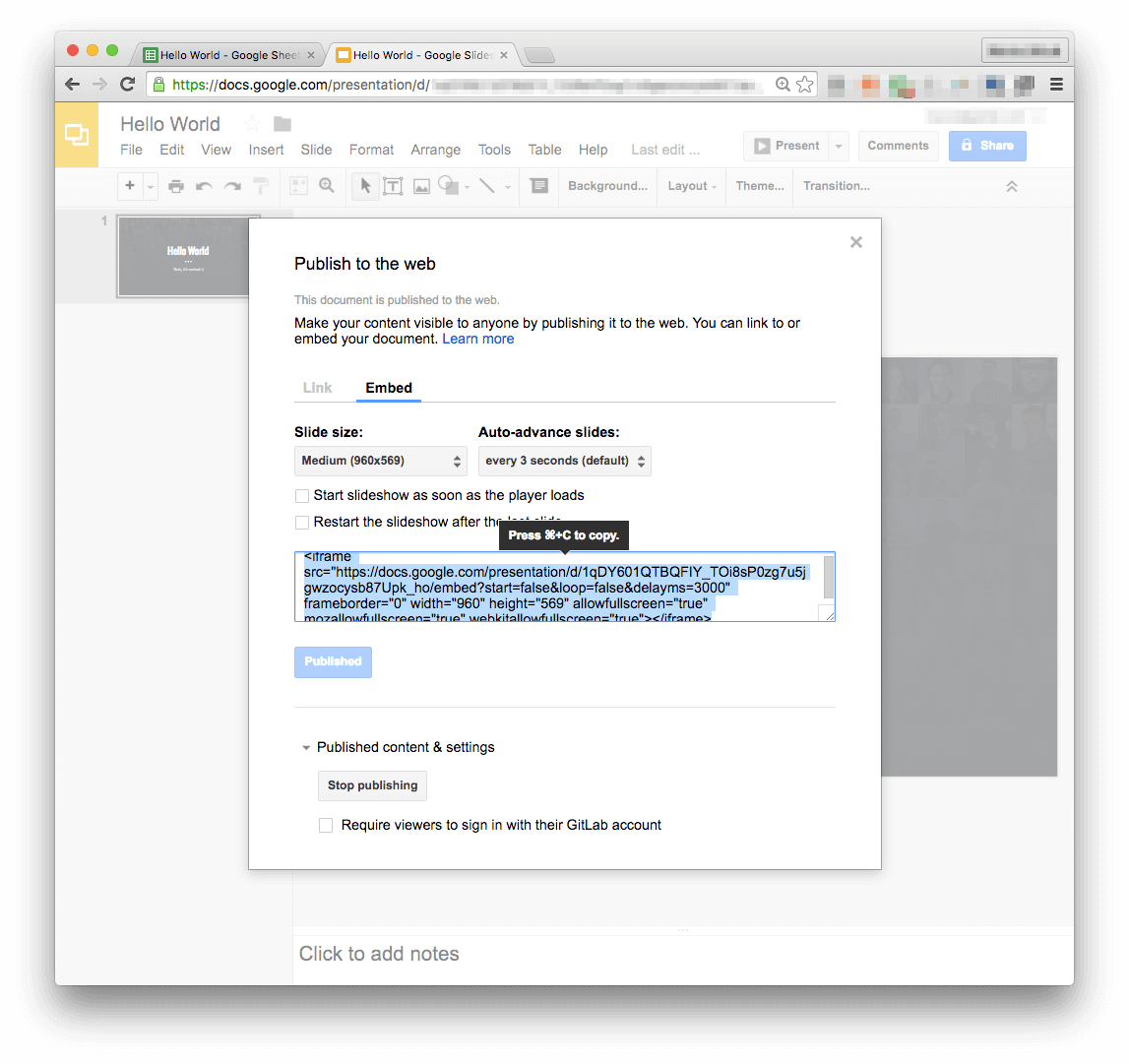
<iframe src="https://docs.google.com/spreadsheets/d/1jAnvYpRmNu8BISIrkYGTLolOTmlCoKLbuHVWzCXJSY4/pubhtml?widget=true&amp;headers=false"></iframe>

</figure>

#### Output:

#### Google Slides

Let's exemplify with this [GitLab deck template](https://docs.google.com/presentation/d/16FZd01-zCj_1jApDQI7TzQMFnW2EGwtQoRuX82dkOVs/edit?usp=sharing). Follow the steps [above](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#embed-documents) to find the iframe:



Copy the code below and paste to your markdown file (leave a blank line above and below it). Then replace the <iframe>with your own:

<figure class="video\_container">

<iframe src="https://docs.google.com/presentation/d/16FZd01-zCj\_1jApDQI7TzQMFnW2EGwtQoRuX82dkOVs/embed?start=false&loop=false&delayms=3000" frameborder="0" width="1280" height="749" allowfullscreen="true" mozallowfullscreen="true" webkitallowfullscreen="true"></iframe>

</figure>

#### Output:

#### Google Docs

Embedding Google Docs is not a recommended practice. Prefer converting your document content to markdown instead. If you need to embed it anyway, follow the same instructions and the same logic as we presented for Google Sheets and Slides, wrapping the <iframe> with a <figure> tag:

<figure class="video\_container">

<iframe src="https://docs.google.com/document/d/1mHhOhvvrz7xgUPyn5VWCNuKgew5MRRGZp761B9prPqs/pub?embedded=true"></iframe>

</figure>

### SlideShare

To embed from SlideShare, go to the document you want to embed and hit the **Share** button located below the slides. Copy the code under **Embed** and place it inside the figure tag.

 Be careful to only include the iframe content and strip anything else. SlideShare will also add some other information in the embed code which you will have to remove, otherwise the markdown page will be broken.

For example, let's say we wanted to include the slides from [Ivan's talk on GitLab Pages](http://www.slideshare.net/creatop/how-to-use-any-static-site-generator-with-gitlab-pages). Copying the embed code and stripping everything else except from the iframe, would result in this:

<figure>

<iframe src="//www.slideshare.net/slideshow/embed\_code/key/EixD8OUMBX65Jy" width="595" height="485" frameborder="0" marginwidth="0" marginheight="0" scrolling="no" style="border:1px solid #CCC; border-width:1px; margin-bottom:5px; max-width: 100%;" allowfullscreen> </iframe>

</figure>

 You can safely omit the <figure> tag since SlideShare's widget is already responsive, but we are showing this that way in order to be consistent with the rest of the handbook.

#### Output:

## Embed Tweets

To add tweets to markdown, copy both <blockquote> and <script> tags from the twitter post and paste it into your file. To make it look much nicer, wrap the script into a div.center (created for this specific purpose).

**Important:** if you used the [HTML parser](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#mix-html--markdown-markup) in your post or page, you'll need to set it to false before the div.

Thanks to [@gitlab](https://twitter.com/gitlab) for joining [@RailsGirlsCluj](https://twitter.com/RailsGirlsCluj)! [pic.twitter.com/NOoiqDWKVY](https://t.co/NOoiqDWKVY)

— RailsGirlsCluj (@RailsGirlsCluj) [October 8, 2016](https://twitter.com/RailsGirlsCluj/status/784847271645028352)

Markup:

{::options parse\_block\_html="false" /}

<div class="center">

<blockquote class="twitter-tweet" data-partner="tweetdeck"><p lang="en" dir="ltr">Thanks to <a href="https://twitter.com/gitlab">@gitlab</a> for joining <a href="https://twitter.com/RailsGirlsCluj">@RailsGirlsCluj</a>! <a href="https://t.co/NOoiqDWKVY">pic.twitter.com/NOoiqDWKVY</a></p>&mdash; RailsGirlsCluj (@RailsGirlsCluj) <a href="https://twitter.com/RailsGirlsCluj/status/784847271645028352">October 8, 2016</a></blockquote>

<script async src="//platform.twitter.com/widgets.js" charset="utf-8"></script>

</div>

For more than one Tweet, copy and paste all the code blocks from Twitter into one div.center:

{::options parse\_block\_html="false" /}

<div class="center">

*<!-- first tweet -->*

<blockquote class="twitter-tweet" data-partner="tweetdeck"><p lang="en" dir="ltr">Thanks to <a href="https://twitter.com/gitlab">@gitlab</a> for joining <a href="https://twitter.com/RailsGirlsCluj">@RailsGirlsCluj</a>! <a href="https://t.co/NOoiqDWKVY">pic.twitter.com/NOoiqDWKVY</a></p>&mdash; RailsGirlsCluj (@RailsGirlsCluj) <a href="https://twitter.com/RailsGirlsCluj/status/784847271645028352">October 8, 2016</a></blockquote>

<script async src="//platform.twitter.com/widgets.js" charset="utf-8"></script>

*<!-- second tweet -->*

<blockquote class="twitter-tweet" data-partner="tweetdeck"><p lang="en" dir="ltr">Thanks to <a href="https://twitter.com/gitlab">@gitlab</a> for joining <a href="https://twitter.com/RailsGirlsCluj">@RailsGirlsCluj</a>! <a href="https://t.co/NOoiqDWKVY">pic.twitter.com/NOoiqDWKVY</a></p>&mdash; RailsGirlsCluj (@RailsGirlsCluj) <a href="https://twitter.com/RailsGirlsCluj/status/784847271645028352">October 8, 2016</a></blockquote>

<script async src="//platform.twitter.com/widgets.js" charset="utf-8"></script>

</div>

## Embed Instagram posts

To embed posts from Instagram, begin by pasting the following code into your file:

{::options parse\_block\_html="false" /}

<div align="center">

Then go to the relevant Instagram post on the web: Go to the post URL if you have it, or you can search for the username on [Instagram.com](https://www.instagram.com/), visit their profile, and then click the post to expand it. Click the more […] button and select "Embed."

Copy the code and paste it below <div align="center">.

Add </div> after the code you pasted from Instagram.

## Embed GitLab Snippets

To embed [GitLab Snippets](https://docs.gitlab.com/ee/user/snippets.html#embedded-snippets) to a markdown file, copy the embed code from your public snippet and paste it in the file.

Markup:

*<!-- leave a blank line here -->*

<script src="https://gitlab.com/gitlab-org/gitlab-ce/snippets/1717978.js"></script>

*<!-- leave a blank line here -->*

Renders:

## Markdown Editors

Please use one of the following code editors to write in markdown, or another **code editor** of your preference.

It is **not** recommend writing your document in a regular text editor, then copy-pasting to markdown, as it most likely will bring some characters with a different encoding (non UTF-8), which will cause the markdown to not render correctly.

In case you don't have a choice and need to import a text already written in a text editor, paste it to your markdown file using **command+shift+V** on a Mac, or **control+shift+V** on Windows or Linux. You might minimize the cause of trouble by pasting without format. But yet, is not guaranteed it is going to work, so double check your HTML output.

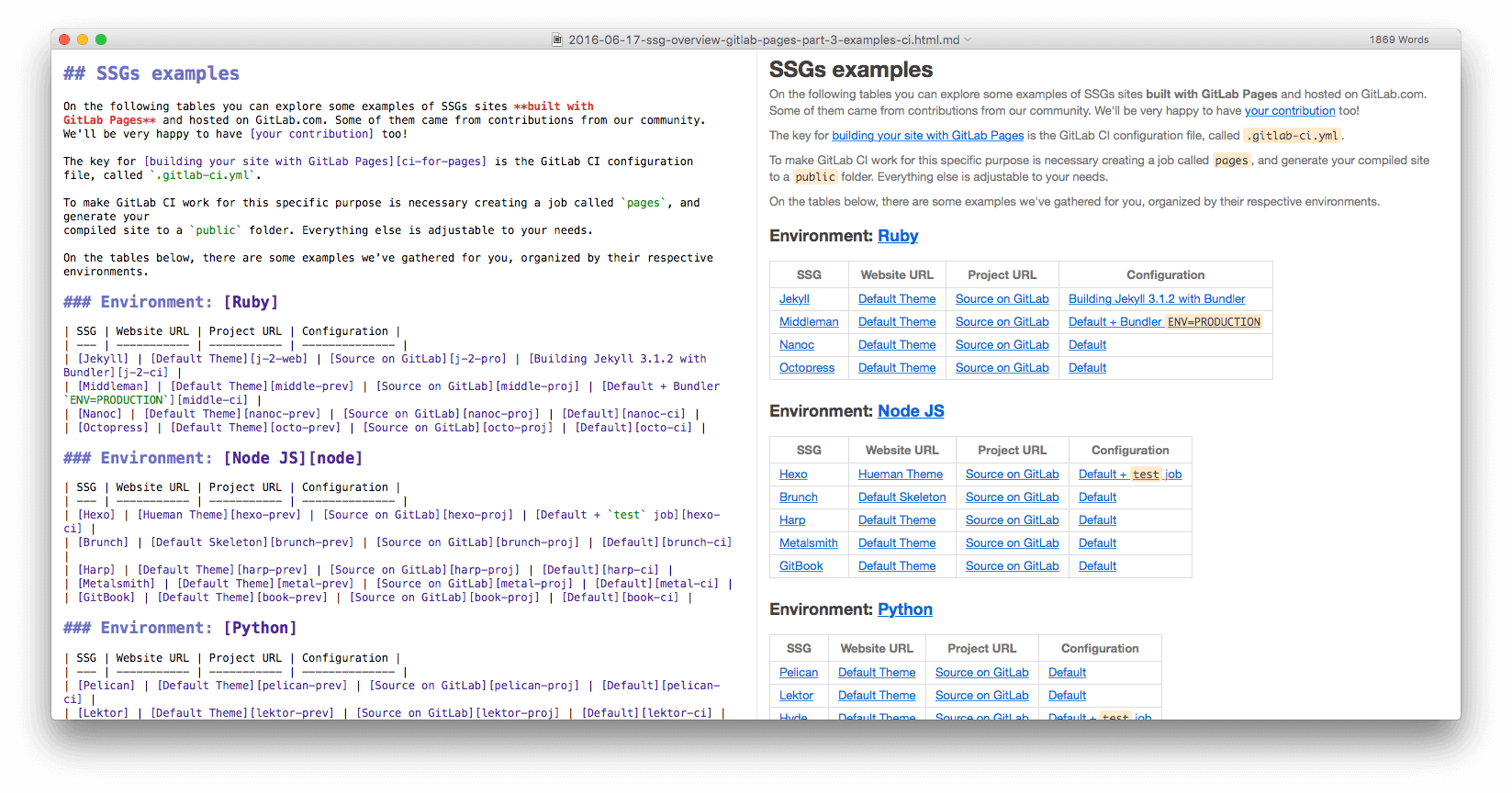
Regular Code Editors

* [Sublime Text 3](https://www.sublimetext.com/3)
* [Atom](https://atom.io/)

Markdown editors (type and preview simultaneously)

* Markdown editors for Mac: [Mou](http://25.io/mou/), [iA Writer](https://ia.net/writer/)
* In-browser markdown editor: [StackEdit](https://stackedit.io/)

If you're not used to writing markdown, those editors can be helpful. Check a screenshot below of a file being edited on Mou. On your left, there's the markdown markup you're writing, and on your right, a preview of the output. The preview won't be exactly as the final result, but it's a very good approximation, which gives you a good idea on what you're doing while you type.



[StackEdit](https://stackedit.io/) is awesome too, you can work on a markdown file even if you're away from your computer, or out of resources. It works from every major browser and automatically saves your work to Google Drive.

## Complementary Notes

* Words must be separated by one single space only. Do not leave more blank spaces than the necessary, they can render differently than expected and can cause other issues.
* Do not leave blank spaces at the end of sentences.
* Always leave a blank line between block-level markup elements, except between list items. Example:
* ---- (markup for horizontal line)
* *<!-- blank line -->*
* Paragraph.
* *<!-- blank line -->*
* Do not leave blank lines within list items:
* *<!-- blank line -->*
* - Item 1
* - Item 2
* - Item 3
* As explained [above](https://about.gitlab.com/handbook/product/technical-writing/markdown-guide/#headings), do **not** skip headings. Always do h1 → h2 → h3 → h4. Never h2 → h4.
* Prefer short titles and headings. Do not punctuate them (unless they require a question mark or an exclamation).
* Try not to punctuate list items, but if you do, be consistent and do that throughout the list.
* If you have to mention a non-clickable URL, prefer using backticks: http://an-example.com.
* To add fancy emojis to your file, click control+cmd+space on your Mac and check the ⭐️ **magic** ⭐ or use a website like [Emoji Finder](https://emojifinder.com/). Do not overuse them, please!
* If you are confused about any markup that you've found in this file, you can check its [raw file](https://gitlab.com/gitlab-com/www-gitlab-com/raw/master/source/handbook/product/technical-writing/markdown-guide/index.html.md) for reference, where you'll be able to see exactly how everything was written to produce the results you are seeing on this page.