Using the content linter

In this article

You can use content linter to check your contributions for errors.

Articles in the "Contributing to GitHub Docs" section refer to the documentation itself and are a resource for GitHub staff and open source contributors.

About the GitHub Docs content linter

Our content linter enforces style guide rules in our Markdown content.

The linter uses `markdownlint` as the framework to perform checks, report flaws, and automatically fix content, when possible. This framework flexibly runs specific rules, gives descriptive error messages, and fixes errors. The GitHub Docs content linter uses several existing `markdownlint` rules and additional custom rules to check the Markdown content in our `content` and `data` directories. Our custom rules implement checks that are either not yet available in the `markdownlint` framework or are specific to GitHub Docs content. Rules check the syntax for both Markdown and Liquid.

Running the GitHub Docs content linter

The GitHub Docs content linter will run automatically on pre-commit, but you can also run it manually.

Automatically run the linter on pre-commit

When you are writing content locally and committing files using the command line, those staged files will automatically be linted by the content linter. Both warnings and errors are reported, but only errors will prevent your commit from completing.

If any errors are reported, your commit will not complete. You will need to fix the reported errors, re-add the changed files, and commit your changes again. Any errors that are reported must be fixed to prevent introducing errors in the content that are in violation of the GitHub Docs style guide. If any warnings are reported, you can optionally choose to fix them or not.

When you are writing content locally, there are several rules that you can fix automatically using the command line. If you want to automatically fix errors that can be fixed, see "Automatically fix errors that can be fixed."

If you are editing a file in the GitHub UI, you will not be able to automatically fix errors or run the linter on a commit, but you will get a CI failure if the content violates any rules with a severity of `error`.

Manually run the linter

Run the linter on staged and changed files

Use the following command to run the linter locally on your staged and changed files. It will output both `warning` and `error` severity flaws.

```
npm run lint-content
```

Run the linter on staged and changed files and only report errors

Use the following command to run the linter locally on your staged and changed files, and report only `error` severity flaws.

```
npm run lint-content -- --errors
```

Run the linter on specific files or directories

Use the following command to run the linter locally on specific files or directories. Separate multiple paths with a space. You can include both files and directories in the same command.

Shell

```
npm run lint-content -- \
  --paths content/FILENAME.md content/DIRECTORY
```

Automatically fix errors that can be fixed

If an error has `fixable: true` in its description, you can use the following commands to automatically fix them.

Run this command to fix staged and changed files only:

```
npm run lint-content -- --fix
```

Run this command to fix specific files or directories:

```
npm run lint-content -- \
  --fix --paths content/FILENAME.md content/DIRECTORY
```

Run a specific set of linter rules

Use the following command to run one or more specific linter rules. These examples run the `heading-increment` and `code-fence-line-length` rules. Replace `heading-increment code-fence-line-length` with one or more linter aliases that you would like to run. To see the list of linter rules you can pass to this option, run `npm run lint-content -- --help`. You can use either the short name (for example, `MD001`) or long name (for example, `heading-increment`) of a linter rule.

Run the specified linter rules on all staged and changed files:

```
npm run lint-content -- \
  --rules heading-increment code-fence-line-length
```

Run the specified linter rules on specific files or directories:

```
npm run lint-content -- \
  --rules heading-increment code-fence-line-length \
  --path content/FILENAME.md content/DIRECTORY
```

Bypass the commit hook

If the linter catches errors that you did not introduce, you can bypass the git commit hook by using the `--no-verify` option when you commit your changes.

```
git commit -m 'MESSAGE' --no-verify
```

Display the help menu for the content linter script

```
npm run lint-content -- --help
```

Linting rules

Each rule is configured in a file in `src/content-linter/style`, which is where the severities of rules are defined.

Errors must be addressed before merging your changes to the `main` branch. Warnings should be addressed but do not prevent a change from being merged into the `main` branch. Most rules will eventually be promoted to errors, once the content no longer has warning violations.

Rule ID	Rule Name(s)	Description	Severity	Tags
MD001	heading-increment, header-increment	Heading levels should only increment by one level at a time	error	headings, headers
MD002	first-heading-h1, first-header-h1	First heading should be a top-level heading	error	headings, headers
MD004	ul-style	Unordered list style	error	bullet, ul
MD009	no-trailing-spaces	Trailing spaces	error	whitespace
MD011	no-reversed-links	Reversed link syntax	error	links
MD012	no-multiple-blanks	Multiple consecutive blank lines	error	whitespace, blank_lines
MD014	commands-show-output	Dollar signs used before commands without showing output	error	code
MD018	no-missing-space-	No space after	error	headings,

MD018	no-missing-space-atx	No space after hash on atx style heading	error	headings, headers, atx, spaces
MD019	no-multiple-space-atx	Multiple spaces after hash on atx style heading	error	headings, headers, atx, spaces
MD022	blanks-around-headings, blanks-around-headers	Headings should be surrounded by blank lines	error	headings, headers, blank_lines
MD023	heading-start-left, header-start-left	Headings must start at the beginning of the line	error	headings, headers, spaces
MD027	no-multiple-space-blockquote	Multiple spaces after blockquote symbol	error	blockquote, whitespace, indentation
MD029	ol-prefix	Ordered list item prefix	error	ol
MD030	list-marker-space	Spaces after list markers	error	ol, ul, whitespace
MD031	blanks-around-fences	Fenced code blocks should be surrounded by blank lines	error	code, blank_lines
MD037	no-space-in-emphasis	Spaces inside emphasis markers	error	whitespace, emphasis
MD039	no-space-in-links	Spaces inside link text	error	whitespace, links
MD040	fenced-code-language	Fenced code blocks should have a language specified	error	code, language
MD042	no-empty-links	No empty links	error	links
MD047	single-trailing-newline	Files should end with a single newline character	error	blank_lines
MD049	emphasis-style	Emphasis style should be consistent	error	emphasis
MD050	strong-style	Strong style should be consistent	error	emphasis
search-replace	todocs-placeholder	Catch occurrences of TODOCS placeholder.	error	
search-replace	docs-domain	Catch	warning	

search-replace	docs-domain	Catch occurrences of docs.gitub.com domain.	warning	
search-replace	help-domain	Catch occurrences of help.github.com domain.	error	
search-replace	preview-domain	Catch occurrences of preview.ghdocs.com domain.	error	
search-replace	developer-domain	Catch occurrences of developer.github.com domain.	error	
search-replace	deprecated liquid syntax: site.data	Catch occurrences of deprecated liquid data syntax.	error	
search-replace	deprecated liquid syntax: octicon-	The octicon liquid syntax used is deprecated. Use this format instead `octicon "<octicon-name>" aria-label="<Octicon aria label>"`	error	
GH001	no-default-alt-text	Images should have meaningful alternative text (alt text)	error	accessibility, images
GH002	no-generic-link-text	Avoid using generic link text like `Learn more` or `Click here`	error	accessibility, links
GHD030	code-fence-line-length	Code fence lines should not exceed a maximum length	warning	code, accessibility
GHD032	image-alt-text-end-punctuation	Alternate text for images should end with punctuation	error	accessibility, images
GHD004	image-file-kebab-case	Image file names must use kebab-case	error	images
GHD033	incorrect-alt-text-length	Images alternate text should be between 40-150 characters	warning	accessibility, images
GHD002	internal-links-no-lang	Internal links must not have a	error	links, url

	lang	must not have a hardcoded language code		
GHD003	internal-links-slash	Internal links must start with a /	error	links, url
GHD031	image-alt-text-exclude-words	Alternate text for images should not begin with words like "image" or "graphic"	error	accessibility, images
GHD034	list-first-word-capitalization	First word of list item should be capitalized	warning	ul, ol
GHD001	link-punctuation	Internal link titles must not contain punctuation	error	links, url
GHD008	early-access-references	Files that are not early access should not reference early-access or early-access files	error	feature, early-access
GHD021	yaml-scheduled-jobs	YAML snippets that include scheduled workflows must not run on the hour and must be unique	error	feature, actions
GHD006	internal-links-old-version	Internal links must not have a hardcoded version using old versioning syntax	error	links, url, versioning
GHD005	hardcoded-data-variable	Strings that contain "personal access token" should use the product variable instead	error	single-source
GHD013	github-owned-action-references	GitHub-owned action references should not be hardcoded	error	feature, actions
GHD016	liquid-quoted-conditional-arg	Liquid conditional tags should not quote the conditional argument	error	liquid, format
GHD014	liquid-data-references-defined	Liquid data or indented data references were found in content	error	liquid

		that have no value or do not exist in the data directory		
GHD015	liquid-data-tag-format	Liquid data or indented data references tags must have the correct number of arguments and spacing	error	liquid, format
GHD010	frontmatter-hidden-docs	Articles with frontmatter property `hidden` can only be located in specific products	error	frontmatter, feature, early-access
GHD009	frontmatter-early-access-references	Files that are not early access should not have frontmatter that references early-access	error	frontmatter, feature, early-access
GH011	frontmatter-video-transcripts	Video transcript must be configured correctly	error	frontmatter, feature, video-transcripts
GHD012	frontmatter-schema	Frontmatter must conform to the schema	error	frontmatter, schema
GHD007	code-annotations	Code annotations defined in Markdown must contain a specific layout frontmatter property	error	code, feature, annotate, frontmatter
GHD017	frontmatter-liquid-syntax	Frontmatter properties must use valid Liquid	error	liquid, frontmatter
GHD018	liquid-syntax	Markdown content must use valid Liquid	error	liquid
GHD019	liquid-if-tags	Liquid `ifversion` tags should be used instead of `if` tags when the argument is a valid version	error	liquid, versioning
GHD020	liquid-ifversion-tags	Liquid `ifversion` tags should contain valid version names as arguments	error	liquid, versioning

Suppressing linter rules

Rarely, you may need to document something that violates one or more linter rules. In these cases, you can suppress rules by adding a comment to the Markdown file. You can disable all rules or specific rules. Always try to limit as few rules as possible. You can disable a rule for an entire file, for a section of a Markdown file, a specific line, or the next line.

For example, if you are writing an article that includes the regular expression `(^|/)[Cc]+odespace/` that checks for reversed link syntax, it will trigger the `MD011` rule that checks for reversed links. You can disable the rule `MD011` on that specific line by adding the following comment.

```
(^|/)[Cc]+odespace/ <!-- markdownlint-disable-line MD011 -->
```

If the line you're trying to ignore is in a code block, you can ignore the code block by surrounding it with the following comments.

````
<!-- markdownlint-disable MD011 -->
```
(^|/)[Cc]+odespace/
```
<!-- markdownlint-enable MD011 -->
````

You can use these comments to enable or disable rules.

Comment	Effect
`<!-- markdownlint-disable -->`	Disable all rules
`<!-- markdownlint-enable -->`	Enable all rules
`<!-- markdownlint-disable-line -->`	Disable all rules for the current line
`<!-- markdownlint-disable-next-line -->`	Disable all rules for the next line
`<!-- markdownlint-disable RULE-ONE RULE-TWO --`	Disable one or more rules by name
`<!-- markdownlint-enable RULE-ONE RULE-TWO -->`	Enable one or more rules by name
`<!-- markdownlint-disable-line RULE-NAME -->`	Disable one or more rules by name for the current line
`<!-- markdownlint-disable-next-line RULE-NAME`	Disable one or more rules by name for the next line