Krik Documentation

This page provides comprehensive documentation for using Krik, the static site generator.

Quick Start

Installation

From Homebrew (macOS/Linux) - Recommended

```
# Install from homebrew tap
brew install mcaserta/krik/krik
```

From Crates.io

```
# Install globally from crates.io
cargo install krik

# No additional setup required - themes and content are embedded!
```

From Source

```
# Clone the repository
git clone <repository-url>
cd krik

# Build the project
cargo build --release

# The executable will be at target/release/kk
```

Getting Started

Initialize a New Site

Create a new Krik site with sample content and default theme:

```
kk init my-blog             # Create new site in 'my-blog' directory
cd my-blog
kk server                   # Start development server
```

Or initialize in the current directory:

```
kk init                     # Initialize in current directory
kk init --force             # Overwrite existing files
```

Create Content

Create new blog posts and pages quickly:

```
kk post "My Great Post"           # Create new blog post
kk post                           # Create with default title "New post"
kk post "Custom Post" -f my-slug  # Custom filename

kk page "About Us"                # Create new page
```

```
kk page                          # Create with default title "New page"
kk page "Contact" -f contact     # Custom filename
```

Development Server

Start the development server with live reload:

```
kk server                    # Start on port 3000 with live reload
kk server --port 8080        # Custom port
kk server --no-live-reload   # Disable live reload (useful for mobile devices)
```

Generate Static Site

```
# Generate site from current directory
kk

# Generate with custom paths
kk --input ./content --output ./_site --theme ./themes/custom
```

Linting Content

Use the linter to validate front matter, filenames, and conventions:

```
kk lint                    # Lint default content directory
kk lint --input content    # Lint a specific directory
kk lint --strict           # Treat warnings as errors
```

What it checks

- Title: required and non-empty
- Language codes: must match filename suffix (e.g., `hello.it.md` → `it`)
- Slugs: filename stem must be slug-like (lowercase, numbers, hyphens)
- Layout: warns on unrecognized values and directory/layout mismatches
- Date: warns if missing for posts; warns if > 1 year in the future
- Tags: array of non-empty strings; warns when tags are not slug-like
- TOC: warns if `toc` is not a boolean
- Duplicate slugs: within the same directory and language
- Duplicate titles: warns within the same directory and language

The command exits non-zero on errors. In `--strict` mode, warnings are also treated as errors.

Content Organization

Directory Structure

```
content/
├── site.toml           # Site configuration
├── posts/              # Blog posts
│   ├── hello.md
│   └── hello.es.md     # Spanish translation
├── pages/              # Static pages
│   └── about.md
└── images/             # Static assets
    └── logo.png
```

Site Configuration

Create a `site.toml` file in your content directory:

```toml
title = "My Website"
base_url = "https://example.com"  # Optional, for feed generation
```

Configuration Options:

- `title`: Site title (displayed in navigation and feeds)
- `base_url`: Base URL for proper Atom feed link resolution

Front Matter

All markdown files can include YAML front matter for metadata:

```yaml
---
title: "Page Title"
date: 2024-01-15T10:30:00Z
layout: post
tags: ["rust", "web", "static-site"]
toc: true
draft: false
custom_field: "custom value"
---
# Your content here
```

Front Matter Fields

Field	Type	Description
`title`	String	Page/post title
`date`	ISO 8601	Publication date (falls back to file mtime)
`layout`	String	Template to use (`post`, `page`, or custom)
`tags`	Array	Tags for categorization
`toc`	Boolean	Enable table of contents generation
`pdf`	Boolean	Enable PDF generation
`draft`	Boolean	Skip file from processing when `true`
Custom fields	Any	Additional metadata accessible in templates

Templates and Layouts

Automatic Template Selection

- **Posts**: Files in `content/posts/` use the `post` template
- **Pages**: Files in `content/pages/` or root use the `page` template
- **Index**: Homepage uses the `index` template

Manual Template Override

Use the `layout` field in front matter:

```
---
title: "Special Page"
layout: custom
---
```

Template Features

- **Post Template**: Tags, "Back to Home" link, language selector, scroll-to-top
- **Page Template**: Clean layout, language selector (if translations available), scroll-to-top
- **Index Template**: Post listing, theme toggle, scroll-to-top

Internationalization (i18n)

Creating Translations

Add language codes to filenames:

- `hello.md` - Default language (English)
- `hello.it.md` - Italian translation
- `hello.es.md` - Spanish translation
- `hello.fr.md` - French translation

Supported Languages

Krik maintains an internal language map used for validation and display names. Out of the box it supports: `en`, `it`, `es`, `fr`, `de`, `pt`, `ja`, `zh`, `ru`, `ar`. Language names in the UI are resolved from this map.

Language Navigation & Index Selection

- Pages with translations automatically show a language selector dropdown.
- The posts index shows one entry per post base name. If multiple language versions exist, the index prefers the default language. If only a non-default language exists (e.g., only `foo.it.md`), that translation is included.

Advanced Features

Table of Contents

Enable TOC generation with `toc: true` in front matter:

```
---
title: "Long Article"
toc: true
---
```

Features:

- Automatic ID generation for headings (AST-based)
- Hierarchical structure preservation
- Clickable navigation links
- Smooth scrolling to sections

Footnotes

Krik supports enhanced footnotes with bidirectional navigation:

```
This has a footnote[^1].

[^1]: This is the footnote content.
```

Features:

- Click footnote numbers to jump to definitions
- Click return arrows (↩️) to return to text
- Smooth scrolling for all footnote navigation

Scroll-to-Top Button

Automatically appears on longer pages with smart visibility:

- Hidden by default until scrolling >300px
- Fixed position in bottom-right corner
- Smooth scrolling animation
- Theme-aware styling
- Mobile-optimized size and positioning

SEO and Discovery

Krik automatically generates SEO-optimized files for search engines and web crawlers:

Atom Feed Generation

Automatically generates an RFC 4287 compliant Atom feed at `feed.xml`:

Features:

- Only includes posts (content with `post` template)
- Limited to 20 most recent posts
- Full HTML content with proper XML escaping
- xml:base support when `base_url` is configured
- Proper metadata (titles, dates, IDs)

XML Sitemap Generation

Automatically generates a comprehensive XML sitemap at `sitemap.xml`:

Features:

- XML Schema validation with proper namespaces
- Multilingual support with `<xhtml:link>` alternate language declarations
- One entry per content piece with canonical URLs (prefers English)
- Proper priority and change frequency settings
- Excludes draft content automatically

robots.txt Generation

Automatically generates SEO-optimized robots.txt:

Features:

- References sitemap.xml location
- Allows all content by default (good for most static sites)
- Blocks access to system files and build directories
- Includes bot-specific rules for major search engines
- Blocks known problematic crawlers/scrapers
- Includes polite crawl delay settings

PDF Generation

Krik supports automatic PDF generation for your content using pandoc and the typst engine. This feature allows you to provide downloadable PDF versions of your posts and pages.

Prerequisites

Before using PDF generation, you need to install the required external tools:

Install pandoc
macOS:

```
brew install pandoc
```

Ubuntu/Debian:

```
sudo apt install pandoc
```

Windows: Download from pandoc.org or use:

```
winget install pandoc
```

Install typst
All platforms:

```
cargo install typst-cli
```

Alternative for macOS:

```
brew install typst
```

Enabling PDF Generation

To enable PDF generation for a document, add `pdf: true` to your front matter:

```
---
title: "My Article"
date: 2025-01-15T10:30:00Z
pdf: true
---
# My Article

This content will be available as both HTML and PDF.
```

Features

Automatic PDF Links

When PDF generation is enabled, Krik automatically adds a PDF download link (📄) to your HTML templates, positioned next to the theme switcher. The link appears only for documents with `pdf: true` in their front matter.

Language-Aware Filenames

PDF files are generated with language-aware filenames:

- `welcome.md` ⟶ `welcome.pdf`
- `welcome.it.md` ⟶ `welcome.it.pdf`
- `about.fr.md` ⟶ `about.fr.pdf`

Conditional Appendix

When your site has a `base_url` configured in `site.toml`, PDFs include an appendix with:

- **Download URL**: Link to the original web version
- **Generation timestamp**: When the PDF was created
- **Multi-language support**: Appendix text translated based on document language

Supported appendix languages: English, Italian, Spanish, French, German, Portuguese, Japanese, Chinese, Russian, Arabic.

Image Path Resolution

Krik automatically resolves relative image paths in PDFs, handling complex patterns like:

- `![Image](content/images/photo.jpg)`
- `![Diagram](assets/diagrams/flow.png)`
- `![Logo](content/pages/logo.svg)`

Configuration

PDF generation works with your existing site configuration. Make sure your `site.toml` includes:

```
title = "My Site"
base_url = "https://mysite.com"  # Optional: enables PDF appendix with download
URL
```

Development Workflow

```
# 1. Add pdf: true to your document
echo '---
title: "My PDF Post"
pdf: true
---

# My PDF Post

Content goes here...' > content/posts/my-post.md
```

```
# 2. Generate site (requires pandoc and typst)
kk

# 3. Check generated files
ls _site/posts/
# Output: my-post.html, my-post.pdf
```

Troubleshooting

Error: "pandoc not found"

- Install pandoc using the instructions above
- Ensure pandoc is in your system PATH

Error: "typst not found"

- Install typst-cli: `cargo install typst-cli`
- Verify installation: `typst --version`

PDF links not appearing in HTML

- Ensure `pdf: true` is set in the document's front matter
- Rebuild your site with `kk`
- Check that the PDF file was generated in the output directory

Images missing in PDF

- Ensure image paths are relative to the content source directory
- Use forward slashes in paths even on Windows
- Verify image files exist at the specified paths

⚠️ Error Handling

Krik uses typed errors for clear diagnostics and actionable messages.

Error Types

- `ConfigError`: configuration files and parsing
- `IoError`: file and directory operations
- `MarkdownError`: front matter and markdown parsing
- `TemplateError`: template compilation and rendering (includes template name)
- `ThemeError`: theme loading and asset handling
- `ServerError`: development server issues
- `ContentError`: content creation and validation failures
- `GenerationError`: site generation pipeline failures

Exit Codes

The CLI maps error types to exit codes:

- Config (2), I/O (3), Markdown (4), Template (5), Theme (6), Server (7), Content (8), Generation (9)

Tips

- Use `-v/--verbose` for detailed logs
- Error messages include paths/template names for faster troubleshooting

Theme System

Light/Dark Mode

Automatic Detection:

- Detects OS theme preference via CSS `prefers-color-scheme`
- Supports all major platforms (Windows, macOS, Linux, iOS, Android)
- Real-time updates when OS theme changes

Manual Toggle:

- Theme button (🌙 /☀️) in top navigation
- Saves preference to localStorage
- Overrides automatic detection
- Smooth 0.3s transitions

Customization

The theme uses CSS custom properties for easy customization:

```
:root {
  --bg-color: #ffffff;
  --text-color: #333333;
  --link-color: #0066cc;
  /* ... more variables */
}
```

Theme Gallery

See all available themes showcased with live previews:

- **Themes Demo**: https://themes.krik.mirkocaserta.com

Command Line Reference

Main Commands

```
kk [OPTIONS]              # Generate static site
kk init [DIR]             # Initialize new site
kk post [TITLE]           # Create new blog post
kk page [TITLE]           # Create new page
kk server [OPTIONS]       # Start development server
```

Global Options

Option	Description	Default
`-i, --input <DIR>`	Input directory	`content`
`-o, --output <DIR>`	Output directory	`_site`
`-t, --theme <DIR>`	Theme directory	`themes/default`
`-h, --help`	Show help	
`-V, --version`	Show version	

Init Command

```
kk init [DIR] [OPTIONS]
```

Option	Description
`[DIR]`	Directory to initialize (default: current directory)
`-f, --force`	Overwrite existing files

Post/Page Commands

```
kk post [TITLE] [OPTIONS]
kk page [TITLE] [OPTIONS]
```

Option	Description	Default
`[TITLE]`	Content title	"New post" / "New page"
`-f, --filename <NAME>`	Custom filename (without .md)	Generated from title
`--content-dir <DIR>`	Content directory path	`content`

Server Command

```
kk server [OPTIONS]
```

Option	Description	Default
`-i, --input <DIR>`	Input directory	`content`
`-o, --output <DIR>`	Output directory	`_site`
`-t, --theme <DIR>`	Theme directory	`themes/default`
`-p, --port <PORT>`	Server port	`3000`
`--no-live-reload`	Disable live reload	Live reload enabled

Generated Output

Krik generates a complete static site with:

- **HTML files**: Preserving directory structure

- **Language variants**: `file.lang.html` for translations
- **Static assets**: Images, CSS, etc. copied as-is
- **Theme assets**: CSS and JavaScript from theme directory
- **Atom feed**: `feed.xml` with proper link resolution
- **XML sitemap**: `sitemap.xml` with multilingual support
- **robots.txt**: SEO-optimized with sitemap reference
- **Navigation**: TOCs, footnote links, scroll-to-top buttons

Example Output Structure

```
_site/
├── index.html          # Homepage
├── feed.xml           # Atom feed
├── sitemap.xml        # XML sitemap
├── robots.txt         # SEO robots file
├── assets/            # Theme assets
│   ├── css/main.css
│   └── js/main.js
├── posts/
│   ├── hello.html
│   └── hello.es.html  # Translation
├── pages/
│   └── about.html
└── images/
    └── logo.png       # Static assets
```

Best Practices

Content Organization

- Use `posts/` for blog entries and time-sensitive content
- Use `pages/` for static pages like About, Contact, etc.
- Keep assets organized in subdirectories
- Use consistent naming conventions for translations

Front Matter

- Always include a `title` for better navigation
- Use `date` for posts to ensure proper chronological ordering
- Add `tags` to posts for better categorization
- Enable `toc` for longer articles with multiple sections

Performance

- Optimize images before adding to content
- Use appropriate image formats (WebP when possible)
- Keep individual posts/pages to reasonable lengths
- Use drafts (`draft: true`) for work-in-progress content

Accessibility

- Use proper heading hierarchy (H1 → H2 → H3)
- Include alt text for images
- Ensure good color contrast in custom themes
- Test with keyboard navigation

Deployment

GitHub Pages

You can automatically deploy your Krik site to GitHub Pages using GitHub Actions. This workflow will build and deploy your site whenever you push to the main branch.

Setup Steps

1. **Create the workflow file**: Add .github/workflows/build-and-deploy.yml to your repository.
2. **Enable GitHub Pages**:
   - Go to your repository settings
   - Navigate to "Pages" section
   - Under "Source", select "Deploy from a branch"
   - Choose the `gh-pages` branch
   - Select "/ (root)" as the folder
   - Click "Save"
3. **Configure your site**: Ensure your content is in the `content/` directory with proper structure
4. **Deploy**: Push to your main branch to trigger the deployment

What the Workflow Does

The GitHub Actions workflow automatically:

- **Installs dependencies**: Sets up Rust toolchain and installs Krik from crates.io
- **Generates the site**: Runs `kk` to build your static site
- **Creates gh-pages branch**: Sets up the deployment branch if it doesn't exist
- **Deploys files**: Copies generated files to the gh-pages branch
- **Adds .nojekyll**: Prevents GitHub from processing files with Jekyll
- **Pushes changes**: Commits and pushes the generated site

Workflow Features

- **Automatic deployment**: Triggers on every push to main branch
- **Manual trigger**: Can be run manually via GitHub Actions interface
- **Branch management**: Handles both new and existing gh-pages branches
- **Clean deployment**: Removes old files before deploying new ones
- **Skip empty deployments**: Only commits when there are actual changes

Repository Structure

Your repository should look like this:

```
your-repository/
├── .github/
│   └── workflows/
│       └── build-and-deploy.yml    # Deployment workflow
├── content/                        # Your Krik content
│   ├── site.toml
│   ├── posts/
│   │   └── *.md
│   └── pages/
│       └── *.md
└── README.md
```

After the first successful deployment, your site will be available at: `https://yourusername.github.io/your-repository-name/`

This documentation covers all major features of Krik. For more examples, check out the other posts and pages in this demo site!

Document Information