

md-pdf

A comprehensive guide



tschinz - 2026-01-22 - 0.1.0

TAGS [user-guide](#) [documentation](#) [md-pdf](#) [markdown to pdf](#) [typst](#) [configuration](#)
[templates](#)

Contents

1	Overview	3
1.1	Key Benefits	3
2	Installation	3
2.1	Prerequisites	3
2.1.1	Rust Toolchain (Optional, for building from source)	3
2.1.2	Typst CLI (Required)	4
2.1.3	Fonts (optional)	4
2.2	Installing md-pdf	4
2.2.1	First Run & Auto-Setup	4
2.2.2	Verification	5
3	Configuration System	5
3.1	Auto-Configuration	5
3.2	Creating Configuration Files	5
3.2.1	Automatic Creation	5
3.2.2	Manual Configuration	5
3.3	Configuration File Locations	6
3.4	Configuration File Content	6
3.5	Configuration Options Reference	6
3.6	Viewing Configuration	7
4	Templates System	7
4.1	Built-in Templates	7
4.1.1	None	7
4.1.2	simple	8
4.1.3	playful	9



4.1.4	brutalist	9
4.1.5	darko	10
4.2	Template Selection Priority	11
4.3	Template Variables & Data Passing	11
4.3.1	System Variables	11
4.3.2	Front Matter Variables	11
4.3.3	Custom Fields	12
4.4	Using Variables in Templates	12
4.4.1	Basic Variable Access	12
4.4.2	Advanced Variable Processing	12
4.4.3	Variable Validation and Defaults	12
4.5	Custom Template Development	13
4.5.1	Creating Custom Templates	13
5	Front Matter Support	13
5.1	What is Front Matter?	13
5.2	Basic Front Matter	13
5.3	Supported Data Types	14
5.3.1	Strings	14
5.3.2	Numbers	14
5.3.3	Booleans	14
5.3.4	Arrays/Lists	15
5.3.5	Objects/Dictionaries	15
5.3.6	Dates and Times	15
5.4	Field Precedence and Smart Defaults	16
5.5	Front Matter Example	16
6	Usage Guide	16
6.1	Basic Commands	16
6.1.1	Essential Operations	17
6.1.2	Information and Configuration Commands	17
6.1.3	Link Checking and Validation	18
6.1.3.1	Sample Output	18



1 | Overview

md-pdf is a powerful, lightweight command-line tool that converts Markdown files into professional PDF documents. Built with Rust and powered by the Typst typesetting system, it provides fast, high-quality document generation.

md-pdf bridges the gap between Markdown's simplicity and PDF's professional presentation. It's designed for developers, writers, researchers, and anyone who needs to create beautiful documents from Markdown source files.



1.1 Key Benefits

- 🚀 **Lightning Fast:** PDF generated with Typst, minimal installation footprint
- 🎨 **Professional Output:** High-quality typography and layout
- ⚙️ **Zero Configuration:** Works out of the box with intelligent defaults
- 🛠️ **Highly Customizable:** Flexible template system and configuration options
- 👁️ **Live Preview:** Real-time PDF updates with watch mode
- 📁 **Auto-Open:** Cross-platform PDF viewing with system default applications
- 🔗 **Link Validation:** Built-in checking for external links to ensure document quality
- 📄 **Rich Metadata:** Comprehensive front matter support

2 | Installation

2.1 Prerequisites

Before installing md-pdf, you need:

2.1.1 Rust Toolchain (Optional, for building from source)

If you plan to build from source:

```
1 # Install rustup if needed
2 curl --proto 'https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
3
4 # Restart shell or source the environment
5 source $HOME/.cargo/env
```

Bash



```
6
7 # Verify installation
8 rustc --version
9 cargo --version
```

Visit rust-lang.org for more details.

2.1.2 Typst CLI (Required)

Typst is the typesetting engine that powers PDF generation:

```
1 # Install via cargo (recommended)
2 cargo install typst-cli
3
4 # Verify installation
5 typst --version
```

[Bash](#)

Visit typst.app for more details.

2.1.3 Fonts (optional)

Some templates use custom fonts. If they are not installed a fallback will be used.

- [Fira Sans and Fira Mono](#)
- [Iosevka](#)

2.2 Installing md-pdf

Install directly from the Git repository using cargo:

```
1 # Install latest stable release
2 cargo install --git https://github.com/tschinz/md-pdf
3
4 # Install specific version/tag
5 cargo install --git https://github.com/tschinz/md-pdf --tag v0.1.0
```

[Bash](#)

2.2.1 First Run & Auto-Setup

After installation, md-pdf is ready to use immediately:

```
1 # Convert your first document (triggers auto-setup)
2 md-pdf README.md
3
4 # Convert and open PDF automatically
5 md-pdf README.md --open
6
7 # Check what was automatically created
8 md-pdf --show-config
9
10 # List available templates
11 md-pdf --list-templates
```

[Bash](#)

On first run, md-pdf automatically creates:

- **Configuration file:** `~/.config/md-pdf/config.ron`
- **Templates directory:** `~/.config/md-pdf/templates/`



- **Template files:** `none.typ`, `simple.typ`, `playful.typ` and `brutalist.typ`

2.2.2 Verification

Verify your installation works correctly:

```
1 # Check version
2 md-pdf --version
3
4 # Test conversion with a simple document
5 echo "# Test Document" > test.md
6 echo "This is a test." >> test.md
7 md-pdf test.md
8
9 # Check if PDF was created
10 ls -la test.pdf
```

Bash

3 | Configuration System

md-pdf uses a sophisticated configuration system that provides intelligent defaults while allowing complete customization.

3.1 Auto-Configuration

The tool automatically configures itself on first use:

- Creates a configuration file with sensible defaults
- Sets up a templates directory with proper paths
- Installs basic template files with full content
- No manual setup required!

3.2 Creating Configuration Files

3.2.1 Automatic Creation

Configuration is created automatically on first use, but you can manually create it:

```
1 # Create default configuration file
2 md-pdf --create-config
```

Bash

This creates `~/.config/md-pdf/config.ron` with:

- Proper templates directory path
- Sensible defaults for all options
- Complete template files (`none.typ` and `simple.typ`)
- Detailed comments explaining each option

3.2.2 Manual Configuration

You can also create configurations in specific locations:

```
1 # Project-specific configuration
2 echo '(default_template: Some("custom"))' > ./md-pdf.ron
3
4 # User-specific override
```

Bash



```
5 mkdir -p ~/.config/md-pdf
6 md-pdf --create-config
```

3.3 Configuration File Locations

md-pdf searches for configuration files in this priority order:

1. **./md-pdf.ron** - Project-specific configuration (highest priority)
2. **<binary-dir>/md-pdf.ron** - Portable installation configuration
3. **~/md-pdf.ron** - User home directory
4. **~/.config/md-pdf.ron** - XDG standard location
5. **~/.config/md-pdf/config.ron** - **Recommended default location**
6. **~/.config/zas/md-pdf.ron** - Workflow integration (lowest priority)

3.4 Configuration File Content

The configuration file uses RON (Rust Object Notation) format, which is human-readable and easy to edit:

```
1 (
2     // Path to templates directory (absolute path recommended)
3     // This directory contains .typ template files
4     templates_dir: Some("/Users/username/.config/md-pdf/templates"),
5
6     // Default template to use when none specified
7     // Available options: "none", "simple", or custom template name
8     default_template: Some("simple"),
9
10    // Default language for documents
11    // Affects typography, hyphenation, and formatting
12    default_language: Some("en"),
13
14    // Generate table of contents by default
15    // Can be overridden per document via front matter
16    default_toc: Some(true),
17
18    // Default author name for documents without explicit author
19    // Used when no author specified in front matter
20    default_author: Some("Your Name Here"),
21 )
```

3.5 Configuration Options Reference

Field	Type	Description	Default Value	Examples
templates_dir	Option<PathBuf>	Directory containing template files	~/.config/md-pdf/templates	Custom template collections
default_template	Option<String>	Template name to use by default	"simple"	"academic", "corporate"



Field	Type	Description	Default Value	Examples
default_language	Option<String>	Document language code	"en"	"de", "fr", "es"
default_toc	Option<bool>	Include table of contents	true	Per-document override available
default_author	Option<String>	Default document author	"User"	Your name or organization

3.6 Viewing Configuration

Check your current configuration:

```
1 # Show all configuration details
2 md-pdf --show-config
3
4 # Output example:
5 # Configuration Information
6 # =====
7 # Templates directory: /Users/username/.config/md-pdf/templates
8 # Default template: simple
9 # Default language: en
10 # Default TOC: true
11 # Default author: Your Name
```

Bash

4 | Templates System

The template system controls the visual appearance and layout of your PDF documents. md-pdf includes built-in templates and supports unlimited custom templates.

4.1 Built-in Templates

4.1.1 None

Minimal template with basic styling



USS Enterprise Tech Brief

Markdown Elements Guide



Lt. Commander Data - 2024-12-20 - NCC-1701-D

Tags

starfleettechnicalmarkdown

Participants

Jean-Luc PicardWilliam RikerData

USS Enterprise Tech Brief

A concise guide to markdown elements aboard the Federation flagship.

Core Systems

Warp Drive

The `warp_core` maintains peak efficiency while *dilithium crystals* power our journey. The `old_conduits` have been upgraded.

Systems Status

- Warp coils aligned
- Antimatter stable
 - Primary containment
 - Backup protocols

Emergency Steps

- Initiate shutdown
- Eject core if needed
- Switch to impulse

Engineering Code

Ship's computer Rust function:

```
fn safe_warp_factor(integrity: f64) -> f64 {
    if integrity > 0.95 { 9.6 } else { 1.0 }
}
```

Use `warp_core::engage()` for FTL travel.

System	Status	Power
Phasers	Online	100%
Shields	Raised	98%

Mission Protocols

"To boldly go where no one has gone before." — Starfleet Charter

Away Team Checklist

- [x] Phaser
- [x] Tricorder
- [] Environmental suit

Key equations: $v = wf^2c$, use `scan_freq = 2.4_GHz`

Live long and prosper.

Starfleet Command • Stardate 47391.2

4.1.2 simple

Clean template with headers and footers

USS Enterprise Tech Brief

Markdown Elements Guide



Lt. Commander Data - 2024-12-20 - NCC-1701-D

Tags

starfleettechnicalmarkdown

Participants

Jean-Luc PicardWilliam RikerData

1 USS Enterprise Tech Brief

A concise guide to markdown elements aboard the Federation flagship.

1.1 Core Systems

1.1.1 Warp Drive

The `warp_core` maintains peak efficiency while *dilithium crystals* power our journey. The `old_conduits` have been upgraded.

1.1.1.1 Systems Status

- Warp coils aligned
- Antimatter stable
 - Primary containment
 - Backup protocols

1.1.1.2 Emergency Steps

- Initiate shutdown
- Eject core if needed
- Switch to impulse

1.2 Engineering Code

Ship's computer Rust function:

USS ENTERPRISE TECH BRIEF | MARKDOWN ELEMENTS GUIDE



```
fn safe_warp_factor(integrity: f64) -> f64 {
    if integrity > 0.95 { 9.6 } else { 1.0 }
}
```

Use `warp_core::engage()` for FTL travel.

1.2.1 Tactical Status

System	Status	Power
Phasers	Online	100%
Shields	Raised	98%

1.3 Mission Protocols

"To boldly go where no one has gone before." — Starfleet Charter

1.3.1 Away Team Checklist

- [x] Phaser
- [x] Tricorder
- [] Environmental suit

Key equations: $v = wf^2c$, use `scan_freq = 2.4_GHz`

Live long and prosper.

Starfleet Command • Stardate 47391.2



4.1.3 playful

Colorful Dieter Rams-inspired design

USS Enterprise Tech Brief

Markdown Elements Guide



LT Commander Data • 2024-12-20 • NCC-1701-D

Tags [starfleet](#) [technical](#) [markdown](#)

Participants [Jean-Luc Picard](#) [William Riker](#) [Data](#)

1 USS Enterprise Tech Brief

A concise guide to markdown elements aboard the Federation flagship.

1.1 Core Systems

1.1.1 Warp Drive

The **warp core** maintains peak efficiency while **dilithium crystals** power our journey. The old conduits have been upgraded.

1.1.1.1 Systems Status

- Warp coils aligned
- Antimatter stable
 - Primary containment
 - Backup protocols

1.1.1.2 Emergency Steps

1. Initiate shutdown
2. Eject core if needed
3. Switch to impulse

USS Enterprise Tech Brief | Markdown Elements Guide



1.2 Engineering Code

Ship's computer Rust function:

```
fn safe_warp_factor(integrity: f64) -> f64 {  
    if integrity > 0.95 { 9.8 } else { 1.0 }  
}
```

Use `warp_core::engage()` for FTL travel.

1.2.1 Tactical Status

System	Status	Power
Phasers	Online	100%
Shields	Raised	98%

1.3 Mission Protocols

"To boldly go where no one has gone before." — Starfleet Charter

1.3.1 Away Team Checklist

- [x] Phaser
- [x] Tricorder
- [] Environmental suit

Key equations: $v = wf^3c$, $use_scan_freq = 2.4_ghz$

Live long and prosper.

Starfleet Command • Stardate 47391.2

LT Commander Data

2024-12-20

2 / 2

4.1.4 brutalist

Raw, stark monospace aesthetic



USS ENTERPRISE TECH BRIEF

MARKDOWN ELEMENTS GUIDE



LT. COMMANDER DATA 2024-12-20 NCC-1701-D

Tags [STARFLEET](#) [TECHNICAL](#) [MARKDOWN](#)

Participants [JEAN-LUC PICARD](#) [WILLIAM RIKER](#) [DATA](#)

1 USS ENTERPRISE TECH BRIEF

A concise guide to markdown elements aboard the Federation flagship.

1.1 CORE SYSTEMS

1.1.1 WARP DRIVE

The **warp core** maintains peak efficiency while **dilithium** crystals power our journey. The old-conduits have been upgraded.

1.1.1.1 Systems Status

- Warp coils aligned
- Antimatter stable
 - Primary containment
 - Backup protocols

1.1.1.2 Emergency Steps

1. Initiate shutdown
2. Eject core if needed
3. Switch to impulse

1.2 ENGINEERING CODE

USS ENTERPRISE TECH BRIEF | MARKDOWN ELEMENTS GUIDE

Ship's computer Rust function:

```
fn safe_warp_factor(integrity: f64) -> f64 {
    if integrity > 0.95 { 1.0 } else { 1.0 }
}
```

Use `warp_core::engage()` for FTL travel.

1.2.1 TACTICAL STATUS

System	Status	Power
Phasers	Online	100%
Shields	Raised	98%

1.3 MISSION PROTOCOLS

"To boldly go where no one has gone before." – Starfleet Charter

1.3.1 AWAY TEAM CHECKLIST

- [x] Phaser
- [x] Tricorder
- [] Environmental suit

Key equations: $v = w/c$, use `scan_freq = 2.4_GHz`

Live long and prosper.

Starfleet Command • Stardate 47391.2

LT. COMMANDER DATA | 2024-12-20

2 / 2

4.1.5 darko

May the dark side be with you

USS Enterprise Tech Brief

Markdown Elements Guide



LT. Commander Data • 2024-12-20 • NCC-1701-D

Tags [starfleet](#) [technical](#) [markdown](#)

Participants [Jean-Luc Picard](#) [William Riker](#) [Data](#)

1 USS Enterprise Tech Brief

A concise guide to markdown elements aboard the Federation flagship.

1.1 Core Systems

1.1.1 Warp Drive

The **warp core** maintains peak efficiency while **dilithium** crystals power our journey. The old conduits have been upgraded.

1.1.1.1 Systems Status

- Warp coils aligned
- Antimatter stable
 - Primary containment
 - Backup protocols

USS ENTERPRISE TECH BRIEF | MARKDOWN ELEMENTS GUIDE



1.1.1.2 Emergency Steps

1. Initiate shutdown
2. Eject core if needed
3. Switch to impulse

1.2 Engineering Code

Ship's computer Rust function:

```
fn safe_warp_factor(integrity: f64) -> f64 {
    if integrity > 0.95 { 1.0 } else { 1.0 }
}
```

Use `warp_core::engage()` for FTL travel.

1.2.1 Tactical Status

System	Status	Power
Phasers	Online	100%
Shields	Raised	98%

1.3 Mission Protocols

"To boldly go where no one has gone before." – Starfleet Charter

1.3.1 Away Team Checklist

- [x] Phaser
- [x] Tricorder
- [] Environmental suit

Key equations: $v = w/c$, use `scan_freq = 2.4_GHz`

Live long and prosper.

Starfleet Command • Stardate 47391.2

LT. Commander Data 2024-12-20 2 / 2



4.2 Template Selection Priority

Templates are chosen in this hierarchical order:

1. **Command-line argument:** `-t template_name` (highest priority)
2. **Front matter:** `template: "template_name"`
3. **Configuration default:** `default_template` setting
4. **System fallback:** "none" template (lowest priority)

Example:

```
1 # Command-line overrides everything
2 md-pdf doc.md -t simple # Uses 'simple' regardless of config/frontmatter
```

Bash

4.3 Template Variables & Data Passing

Templates receive comprehensive data from the conversion process. Understanding these variables is crucial for creating custom templates.

4.3.1 System Variables

These variables are automatically provided by md-pdf:

Variable	Type	Description	Example Value
filepath	String	Source markdown file path	"/path/to/document.md"
default_author	String	Author from configuration	"Your Name"
language	String	Document language	"en", "de", "fr"
toc	String	Table of contents flag	"true" or "false"
has_frontmatter	String	Front matter presence indicator	"true" or "false"

4.3.2 Front Matter Variables

All YAML front matter fields become template variables. All fields are optional. Common fields are:

Field	Variable Name	Type	Example Value	Usage
title	title	String	"My Document"	Document title
subtitle	subtitle	String	"Guide"	Document subtitle
author	author	String	"John Doe"	Document author
logo	logo	String	"/logo.png"	Rectangular logo
date	date	String	"2024-01-22"	Publication date
version	version	String	"1.0.0"	Document version
tags	tags	Array	["docs", "guide"]	Document tags
participants	participants	Array	["Him", "Her"]	List of participants
language	language	String	"en"	Document language [en, de, fr]
template	template	String	"brutalist"	Template name [brutalist, playful, simple, none, your custom templatenam]



4.3.3 Custom Fields

Any YAML field in front matter becomes a template variable:

```
1 ---
2 # Custom fields become template variables
3 client: "ACME Corporation"
4 contract_number: "CON-2024-001"
5 review_date: "2024-02-01"
6 budget: 50000
7 approved: true
8 reviewers:
9   - "Alice Johnson"
10  - "Bob Smith"
11 ---
```

YAML

These become available as:

- `sys.inputs.client` → "ACME Corporation"
- `sys.inputs.contract_number` → "CON-2024-001"
- `sys.inputs.budget` → 50000
- `sys.inputs.approved` → "true"
- `sys.inputs.reviewers` → ["Alice Johnson", "Bob Smith"]

4.4 Using Variables in Templates

Templates use Typst syntax to access and manipulate variables:

4.4.1 Basic Variable Access

```
1 // Access system variables
2 #if sys.inputs.toc == "true" [
3   #outline(title: "Table of Contents", depth: 3)
4   #pagebreak()
5 ]
6
7 // Use author with configuration fallback
8 #let doc_author = sys.inputs.at("author", default: sys.inputs.default_author)
```

Typst

4.4.2 Advanced Variable Processing

```
1 // Process arrays (tags, keywords, etc.)
2 #if "tags" in sys.inputs [
3   *Tags: * #sys.inputs.tags.join(", ")
4 ]
```

Typst

4.4.3 Variable Validation and Defaults

```
1 // Validate required fields
2 #let required_fields = ("title", "author", "version")
3 #for field in required_fields [
4   #if field not in sys.inputs [
5     #text(fill: red)[Error: Required field '#field' missing from front matter]
6   ]
```

Typst



```
7 ]
8
9 // Provide smart defaults
10 #let doc_title = sys.inputs.at("title", default: "Untitled Document")
11 #let doc_author = sys.inputs.at("author", default: sys.inputs.default_author)
12 #let doc_version = sys.inputs.at("version", default: "1.0")
```

4.5 Custom Template Development

4.5.1 Creating Custom Templates

1. Navigate to templates directory:

```
1 cd ~/.config/md-pdf/templates
```

Bash

2. Create new template file:

```
1 touch my-custom-template.typ
```

Bash

3. Add template description (recommended):

```
1 // My Custom Academic Template
2 // Professional template for academic papers and research documents
3 // Supports: citations, figures, academic formatting, multiple authors
4 // Created: 2024-01-22
5 // Last modified: 2024-01-22
6
7 // Your template code here...
```

Typst

4. Use your template:

```
1 md-pdf document.md -t my-custom-template
```

Bash

5 | Front Matter Support

Front matter provides document metadata and per-document configuration through YAML headers at the beginning of Markdown files.

5.1 What is Front Matter?

Front matter is a YAML block at the very beginning of a Markdown file, enclosed by triple dashes (—). It contains metadata and configuration options that control how the document is processed and formatted.

5.2 Basic Front Matter

The simplest front matter includes core document information:

```
1 —
2 title: "My Document"
3 author: "John Doe"
4 date: "2024-01-22"
5 tags:
```

YAML



```
6   - tag1
7   - tag2
8   —
9 # Document content starts here...
```

5.3 Supported Data Types

md-pdf handles all standard YAML data types with intelligent processing:

5.3.1 Strings

```
1 # Simple strings
2 title: "Document Title"
3 description: "Single quoted string"
4
5 # Multi-line strings (preserve line breaks)
6 abstract: |
7     This is a multi-line string
8     that preserves line breaks.
9     Perfect for abstracts, descriptions,
10    or detailed explanations.
11
12 # Folded strings (join lines with spaces)
13 summary: >
14     This is a folded string that joins
15     multiple lines with spaces, creating
16     a single paragraph of text.
17
18 # Escaped strings
19 special_chars: "Quotes: \"Hello\", Backslash: \\"
```

YAML

5.3.2 Numbers

```
1 # Integers
2 version_number: 2
3 revision: 3
4 page_count: 42
5 budget: 50000
6
7 # Floating point
8 price: 29.99
9 rating: 4.5
10 percentage: 95.7
```

YAML

5.3.3 Booleans

```
1 toc: true
2 draft: false
3 confidential: true
4 approved: false
5 published: true
```

YAML



5.3.4 Arrays/Lists

```
1 # Simple arrays
2 tags: ["markdown", "pdf", "documentation"]
3 languages: ["en", "es", "fr"]
4
5 # Multi-line arrays
6 keywords:
7   - technical writing
8   - automation tools
9   - document processing
10  - workflow optimization
11
12 # Mixed type arrays
13 scores: [95, 87.5, "excellent", true]
14
15 # Nested arrays
16 matrix:
17   - [1, 2, 3]
18   - [4, 5, 6]
19   - [7, 8, 9]
```

YAML

5.3.5 Objects/Dictionaries

```
1 # Simple objects
2 contact:
3   name: "John Doe"
4   email: "john@example.com"
5   phone: "+1-555-0123"
6
7 # Nested objects
8 project_info:
9   name: "Documentation Project"
10  timeline:
11    start: "2024-01-01"
12    end: "2024-06-30"
13  budget:
14    allocated: 100000
15    spent: 75000
16  team:
17    lead: "Jane Smith"
18    members: ["Alice", "Bob", "Charlie"]
```

YAML

5.3.6 Dates and Times

```
1 # ISO 8601 dates
2 date: "2024-01-22"
3 created: 2024-01-22T10:30:00Z
4 deadline: "2024-12-31T23:59:59Z"
5
6 # Natural language dates (processed as strings)
7 review_date: "January 31, 2024"
```

YAML



```
8 next_meeting: "first Monday of February"
```

5.4 Field Precedence and Smart Defaults

md-pdf uses intelligent precedence for configuration values:

1. **Front matter** (highest priority) - Values explicitly set in YAML
2. **Configuration file** - Defaults from your config file
3. **System defaults** (lowest priority) - Built-in fallbacks

5.5 Front Matter Example

```
1 ---
2 title: "Machine Learning Applications in Document Processing"
3 subtitle: "A Comprehensive Survey and Future Directions"
4 author: "Dr. Sarah Johnson"
5 affiliation: "University of Technology"
6 department: "Computer Science"
7 date: "2024-01-22"
8 template: "academic"
9 toc: true
10 language: "en"
11 tags: ["machine learning", "document processing", "natural language processing", "computer vision",
12       "automation"]
12 abstract: |
13     This paper presents a comprehensive survey of machine learning
14     techniques applied to automated document processing tasks,
15     including natural language processing, computer vision, and
16     hybrid approaches. We analyze current methodologies, identify
17     key challenges, and propose future research directions.
18 keywords:
19   - machine learning
20   - document processing
21   - natural language processing
22   - computer vision
23   - automation
24 ---
```

YAML

6 | Usage Guide

6.1 Basic Commands

md-pdf provides a clean, intuitive command-line interface designed for efficiency and ease of use.

```
1 > md-pdf -h
2 Convert markdown files to PDF using typst
3
4 Usage: md-pdf [OPTIONS] [INPUT]
5
6 Arguments:
7   [INPUT] Path to the input Markdown file
```

Bash



```
8
9 Options:
10 -o, --output <OUTPUT>      Path to the output PDF file
11 -t, --template <TEMPLATE>  list the templates and select the one you want [default: none]
12 -w, --watch                 Watch the input file for changes and rebuild automatically
13     --check-links           Check all links in the markdown file and display warnings for unreachable
                             links
14     --list-templates        List all available templates
15     --create-config          Create default configuration file
16     --show-config            Show configuration file locations and settings
17     --open                  Open the generated PDF file after creation
18 -h, --help                  Print help
19 -V, --version                Print version
```

6.1.1 Essential Operations

```
1 # Convert with all defaults (uses config settings)
2 md-pdf document.md
3
4 # Convert and open PDF automatically
5 md-pdf document.md --open
6
7 # Specify output filename
8 md-pdf document.md -o report.pdf
9
10 # Use specific template (overrides config default)
11 md-pdf document.md -t simple -o formatted-report.pdf
12
13 # Convert with template and auto-open
14 md-pdf document.md -t simple --open
15
16 # Watch file for changes (live preview mode)
17 md-pdf --watch document.md
18
19 # Watch and auto-open after each rebuild
20 md-pdf --watch document.md --open
21
22 # Watch with custom output and template
23 md-pdf --watch document.md -t simple -o live-preview.pdf
24
25 # Watch with custom output, custom template, auto-open, and link checking
26 md-pdf --check-links --watch document.md -t simple --open -o live-preview.pdf
```

Bash

6.1.2 Information and Configuration Commands

```
1 # List all available templates with descriptions
2 md-pdf --list-templates
3
4 # Show current configuration and all paths
5 md-pdf --show-config
6
7 # Create default configuration file (if not exists)
```

Bash



```
8 md-pdf --create-config
9
10 # Show help information
11 md-pdf --help
12
13 # Show version information
14 md-pdf --version
```

6.1.3 Link Checking and Validation

md-pdf includes built-in link checking functionality to validate external links in your markdown documents before PDF conversion.

```
1 # Check all external links in document
2 md-pdf --check-links document.md
```

Bash

6.1.3.1 Sample Output

```
1 🔍 Checking 5 links in 'research-paper.md'...
2 ✅ https://github.com/typst/typst
3 ✅ https://www.rust-lang.org
4 ❌ https://broken-example.invalid - Error: dns error: failed to lookup address information
5 ✅ https://docs.rs
6 ✅ https://wikipedia.org
7
8 📄 Link check summary:
9   ✅ Successful: 4
10  ❌ Failed: 1
11 ⚠️ Some links are not reachable. Consider reviewing them.
12
13 Using template: simple
14 ✓ Successfully converted 'research-paper.md' to 'research-paper.pdf'
```