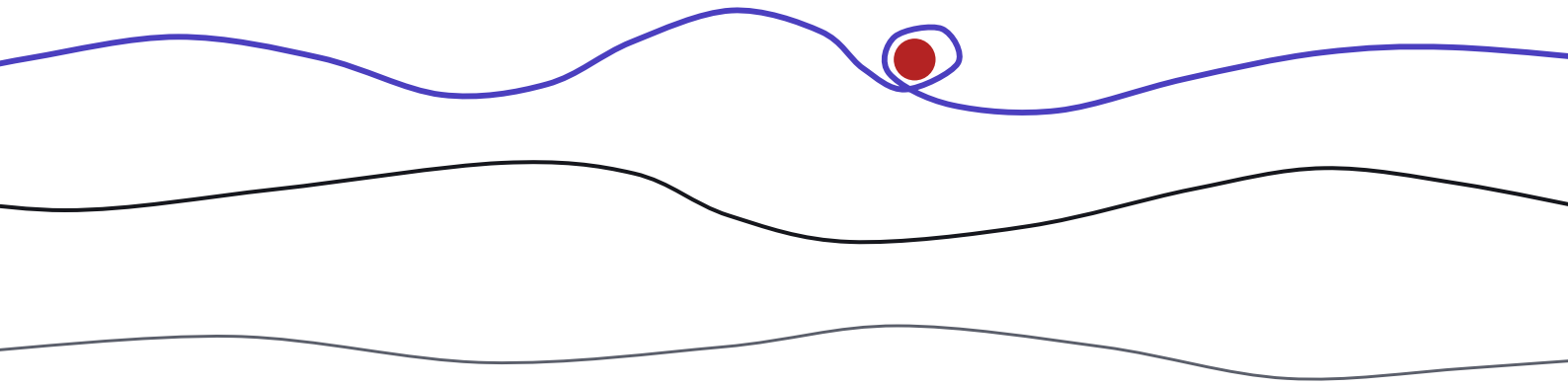
jupiter.au

Jupiter documentation

Documentation of every jupiter.au project in one place. Each project keeps its pages in its own repository; this site reads them from the project's latest release.



DATE

25 November 2026

AUTHOR

belikh

SOURCE

github.com/belikh/docs

Contents

Write

01 **Add a project** 3

01.1 Register a project 3

01.2 What the project's config holds 4

01.3 Sections 4

01.4 Several PDFs 5

01.5 Addresses and links 6

01.6 When a project has errors 6

01.7 When a page moves 6

01.8 Rebuild after a release 6

01.9 Projects without a docs-src folder 6

02 **Writing documentation** 7

02.1 Folder layout 7

02.2 Page header 8

02.3 Directives 8

02.4 Links 8

02.5 Code 9

02.6 Check and build 9

02.7 Pages from other projects 9

02.8 The design system 10

02.9 What a good page has 10

02.10 Voice 10

02.11 Add a graphic 11

02.12 Add a diagram 11

03 **Glossary** 12

JUPITER DOCUMENTATION

Jupiter documentation

Each project writes its pages in a `docs-src` folder in its own repository. This site fetches them from the project's latest release once a day and shows them under the project's name. Add a project has the steps to put a project here, and Writing documentation is the format reference.

SECTION

Write

How documentation is written, registered and published.

01

Add a project

Put a project's documentation on this site while its pages stay in the project's own repository.

A *Project* writes its pages in its own repository. This repository holds no copy of them: `jupiter-docs import` fetches the pages before every build, and the site, the PDF and `llms.txt` are generated from what it fetched.

01.1

Register a project

A public repository of `belikh` registers itself. Commit these two things to its default branch:

IN THE PROJECT'S REPOSITORY

```
docs.config.json        the project name and one sentence about it
docs-src/
  index.md              the project's landing text
  start/install.md      pages, grouped in folders
```

`jupiter-docs init` creates both. The next build of this site finds the repository through `discover` in `docs.config.json` here and adds the project. The site builds once a day, on every push to this repository and on request.

A repository of another owner, or one that should use a fixed branch or another id, gets one line in `sources`:

DOCS.CONFIG.JSON OF THIS REPOSITORY

```
"discover": { "owner": "belikh", "exclude": ["scratch"] },
"sources": [
  { "repo": "someone/their-project" },
  { "repo": "belikh/aquila", "ref": "main", "id": "aquila", "icon": "server" }
]
```

KEY	MEANING
`repo`	`owner/name` on GitHub.
`ref`	A tag, a branch or `latest`. `latest` is the newest release, or the default branch when the project has no release or the release has no `docs.config.json`.
`id`	The first part of the project's addresses, for example `/aquila/`. Default: `id` in the project's own `docs.config.json`, then the repository name in lower case.
`title`, `summary`, `icon`	Replace the values from the project's own `docs.config.json`.
`path`	A local folder in place of `repo`, to preview a project that is not pushed yet.

Private repositories, forks and archived repositories are not discovered.

01.2

What the project's config holds

DOCS.CONFIG.JSON OF THE PROJECT

```
{
  "project": "Aquila",
  "id": "aquila",
  "description": "A fleet manager for NixOS hosts.",
  "icon": "server",
  "versionFrom": "package.json",
  "repo": "https://github.com/belikh/aquila",
  "glossary": { "Host": "One machine that Aquila manages." }
}
```

`project` and `description` become the title and the summary of the project's card on the home page. `icon` is the icon on that card. `id` is the first part of the project's addresses; choose it once, because addresses are permanent. The glossary belongs to the project: `[[term:Host]]` on one of its pages links to the project's own glossary page.

01.3

Sections

A folder in `docs-src` is a section. These five folder names have a fixed title, summary and icon, and appear in this order in every project, so a reader finds the same things in the same place:

FOLDER	TITLE	WHAT GOES IN IT
`start`	Start	Install the project and run it for the first time. One page is often enough.
`guides`	Guides	One task per page, as numbered steps with the commands to copy.
`reference`	Reference	Every command, option, file format and API, in tables.

FOLDER	TITLE	WHAT GOES IN IT
`concepts`	Concepts	How the project works and why it is built this way.
`develop`	Develop	Build, test and release the project.

Any other folder becomes a section named after the folder. A project that needs its own titles or order lists `sections` in its `docs.config.json`; declared sections come first.

01.4

Several PDFs

A project has one PDF that holds every section. A project that needs separate ones, such as a user guide, an installation guide and a developer reference, lists `documents` in its `docs.config.json`:

DOCS.CONFIG.JSON OF THE PROJECT

```
"documents": [
  { "id": "user", "title": "User guide", "summary": "Run Aquila day to day.",
  "sections": ["guides", "concepts"] },
  { "id": "install", "title": "Installation guide", "sections": ["start"] },
  { "id": "developer", "title": "Developer reference", "sections": ["reference",
  "develop"] }
]
```

KEY	MEANING
`id`	Lower case letters, digits and `-`. The PDF is `<project id>-<document id>.pdf`, so this example builds `aquila-user.pdf`, `aquila-install.pdf` and `aquila-developer.pdf`.
`title`	Shown below the project name on the cover, and on the download button.
`summary`	Optional. One sentence, shown when the pointer is over the download button.
`sections`	The section ids in the document, in reading order. A section can be in more than one document.

The project's landing page has one download button per document. The PDF link at the top of a page opens the document that holds the page's section. The project's `index.md` opens every document. A link to a page of another document is plain text in the PDF, because that page is not in the file.

`jupiter-docs check` reports a document that names an unknown section as an error. It warns about a section that is in no document, because its pages are then in no PDF.

> WARNING
>
> A project with `documents` has no `<project id>.pdf`. A link to that file from another site stops working at the next build.

01.5

Addresses and links

A page is published at `/<project id>/<path of the file>/`, so `docs-src/guides/backup.md` of `aquila` is at `/aquila/guides/backup/`. The project's `index.md` is its landing page at `/aquila/`.

`[[guides/backup]]` on a page of the project names a page of the same project. To link to another project, start with its id: `[[design/consuming]]`, or `[[design]]` for its landing page. Relative links to files that are not pages, and images, point at the project's repository at the imported ref.

01.6

When a project has errors

Run `jupiter-docs check` in the project before a release. If a released project still fails the check here, the site is built without it and the build log has a warning with every error. Errors in the pages of this repository stop the build.

Wording is treated differently. A voice finding in a project's page is an error in the project's own check and a warning here. The page stays on the site, and the build log names the sentence to fix (see Voice).

01.7

When a page moves

Addresses are permanent once published. When a file is renamed or a project changes its id, add the old address to `redirects` in `docs.config.json` of this repository. The site keeps a small page there that sends the reader to the new one:

DOCS.CONFIG.JSON OF THIS REPOSITORY

```
"redirects": { "consuming": "design/consuming" }
```

01.8

Rebuild after a release

The daily build picks up a new release. To publish sooner, send a `repository_dispatch` event of type `docs-changed` from the project's release workflow:

AFTER THE RELEASE STEP

```
gh api repos/belikh/docs/dispatches -f event_type=docs-changed
```

The call needs a token that can write to `belikh/docs`.

01.9

Projects without a docs-src folder

Hand-written Markdown files of a repository can be listed one by one. The entry then carries what the project's config would hold:

DOCS.CONFIG.JSON OF THIS REPOSITORY

```
{
  "repo": "belikh/design", "id": "design", "title": "Design system",
  "sections": [{ "id": "use", "title": "Use" }],
  "files": {
    "docs/consuming.md": { "id": "consuming", "title": "Depending on the design
    system", "summary": "How a project takes the design system as a dependency.",
    "section": "use", "order": 10 }
  }
}
```

The import removes each file's first # title and a leading comment, and turns links between the listed files into page links.

SEE ALSO

- Writing documentation
- Glossary

02

Writing documentation

How to write the one source folder that becomes the docs site, the Markdown files in the repository and a PDF.

Documentation is written once, in `docs-src/`, as Markdown with a small header on each file. The `jupiter-docs` command (package `@belikh/jupiter-docs`) turns that folder into three outputs. Never edit an output by hand: the next build overwrites it.

OUTPUT	WHERE	COMMAND
Static site for docs.jupiter.au	`dist/site`	`jupiter-docs build --site`
Plain GitHub Markdown	`docs/*.md`	`jupiter-docs build --md`
PDF, one per project or one per document	`dist/<id>.pdf`	`jupiter-docs build --pdf`

02.1

Folder layout

DOCS-SRC AND DOCS.CONFIG.JSON

```
docs.config.json        project name, description, version, repository, glossary
docs-src/
  index.md              landing page text
  start/install.md      one page per file; the path is the page id: start/install
  guides/backup.md      a folder is a section: this page is in Guides
```

A page belongs to the section named by its folder. The folders `start`, `guides`, `reference`, `concepts` and `develop` have a fixed title and order (see Sections). A page that is not in a folder names its section in its header, and `docs.config.json` lists such sections under `sections`. Pages in a section are sorted by `order`, then by title.

DOCS.CONFIG.JSON

```
{
  "project": "My project",
  "versionFrom": "package.json",
  "repo": "https://github.com/me/my-project",
  "description": "What the project does, in one sentence.",
  "glossary": { "Token": "A named design constant." }
}
```

02.2

Page header

Every page starts with a YAML block. `title` and `summary` are required. `section` is required only on a page that is not in a folder, except `index.md`.

DOCS-SRC/GUIDES/SETUP.MD

```
---
title: Set up the project
summary: Install the tools and run the first build.
order: 20
icon: terminal
audience: developers
status: stable
---
```

`icon` is a name from the icon tier of the graphics set. `audience` is one of users, developers, agents or maintainers. `status` is draft, stable or deprecated. The page title is the top heading, so the body starts at ## and never skips a level.

02.3

Directives

Block directives start with colons. Use four colons for the outer block when you nest one inside another.

CALLOUTS

```
::: note
Plain information.
:::

::: warning Check the version
A title is optional. The kinds are note, tip, warning and danger.
:::
```

> TIP
>
> Callouts become GitHub alerts in the repository Markdown, boxed text on the site and `jp-callout` boxes in the PDF.

One-line directives start with two colons.

GRAPHICS, DIAGRAMS AND THE GLOSSARY

```
::graphic rule/end
::graphic spot/summit A mountain with a flag on the summit
::diagram wiring/board-to-monitor The board's GPIO 4 goes to pin 7 of the monitor
::glossary
```

`::graphic tier/name` places a graphic from `@belikh/jupiter-graphics`. Text after the name is the alt text; leave it out for a decorative graphic. The site draws it inline in the theme colour, the repository Markdown uses a light and dark `<picture>`, and the PDF uses a coloured copy. `::diagram name Alt text` places a drawing that your project made itself, see Add a diagram. `::glossary` lists every term from the glossary in `docs.config.json`.

02.4

Links

Link to another page with its id, and to a heading with its anchor. Plain relative links to `.md` files also work.

CROSS-REFERENCES

```
See [[guides/setup]] or [[guides/setup#install|the install steps]].
A [[term:Token]] is defined once in the glossary.

::: see-also
- [[reference/cli]]
:::
```

A page id names a page of the same project first. Start with a project id to link to another project: `[[design/consuming]]`. A *Token* is a named design constant.

Links to files outside the source folder, such as `../AGENTS.md`, are checked against the repository and become repository links on the site. Raw HTML is not allowed.

02.5

Code

Give every code block a language, and a title when the block is a file or a command with context. Use `text` for plain output.

CODE WITH A TITLE

```
```sh title="Install"
npm install
```
```

02.6

Check and build

COMMANDS

```
jupiter-docs check              # validate the source and its wording
jupiter-docs voice README.rst   # check the wording of any other Markdown or .rst
file
jupiter-docs build              # write docs/*.md and dist/site
jupiter-docs build --pdf        # build the PDFs: one per project, or one per
document (needs xelatex)
jupiter-docs build --check      # fail when docs/*.md is out of date (CI)
jupiter-docs import             # fetch the projects named by sources and discover
(docs-src/_imported)
jupiter-docs init               # scaffold docs-src in a new project
```

`check` names the file and line of each problem:

- a missing title or summary, or an unknown section, graphic, icon or glossary term
- a diagram that is missing, has no alt text, or breaks the drawing rules
- a broken page link or anchor
- a heading that skips a level, an unclosed directive, a code block without a language
- wording that fails the voice rules (see Voice)

> WARNING
>
> `build --md` only writes files that carry the generated header. Hand-written files in `docs/` are never touched or deleted.

02.7

Pages from other projects

A project's pages stay in its repository. `jupiter-docs import` fetches every project named by `sources` and `discover` in `docs.config.json` at its latest release and writes the pages to `docs-src/_imported/<project id>/`. The folder is not committed and the pages are not written to `docs/`. Add a project has the steps and the keys.

02.8

The design system

The site and the PDF take their colours, fonts, graphics and LaTeX classes from the latest `belikh/design` release. `npm run update:design` downloads it into `vendor/design`, which is not committed, and the workflows do the same before every build. When the design system changes, the next build looks different with no change in this repository.

02.9

What a good page has

- A title that names the task or the thing, and a one-sentence summary that says what the reader gets.
- The command or the result in the first screen. Steps are a numbered list, one action each, with the command to copy.
- Every option, flag and file in a table with its default. A reader should not need the source code.
- One working example that can be pasted and run, with the output it prints.
- The errors a reader will meet, quoted exactly, each with its cause and its fix.
- Links to the next page a reader needs, in a `:::see-also` block.

02.10

Voice

A reader who has never seen the project should find the answer on the first screen and understand every sentence on the first read.

- Answer first. The first sentence says what the thing is or what the reader gets.
- One idea per sentence, most under 25 words. Say “you”, use the present tense, write steps as commands.
- State facts a reader can check: numbers, names, defaults, paths, exact error text.
- Use the short word: use, help, show, before, because, can.
- The page does not describe itself, and it stops when the content ends.

`jupiter-docs check` tests the prose of every page. Code blocks, inline code and block quotes are left out, so quoted text stays as given.

FINDING	LEVEL
Praise words such as `powerful`, `robust`, `seamless`; sincerity words such as `simply`, `truly`; signposting such as `worth noting`, `in conclusion`, `moreover`; `not just X but Y`; `this page explains`; an em dash; an emoji	Error
`note that`, `in order to`, `is able to`, `prior to`, `easily`, `X, not Y`, an exclamation mark, a sentence over 35 words	Warning

Fix the sentence when a word is flagged: state the fact the word stood for. `jupiter-docs voice README.rst` runs the same check on any Markdown or `.rst` file. The full guide, with rewrites, is VOICE.md. The visual rules are in `DESIGN.md` of the `jupiter-design-system` skill; the site and the PDF already follow them.

02.11

Add a graphic

Graphics come only from `@belikh/jupiter-graphics`. If the set lacks one, add it to `packages/graphics` in the design repository (`belikh/design`) first. Once that is released, the next docs build can use its `tier/name`. `jupiter-docs check` fails on a name that is not in the set.



SEE ALSO

- Add a project
- Glossary

02.12

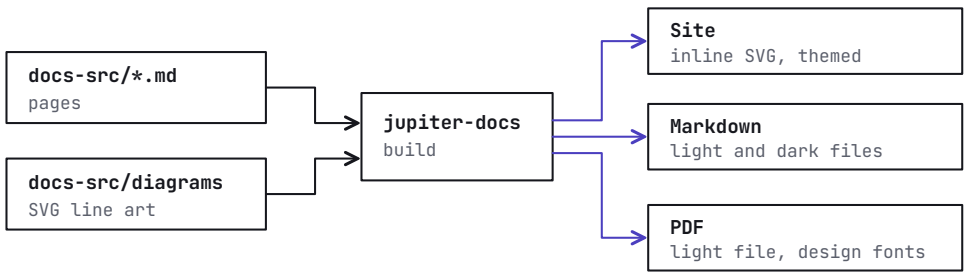
Add a diagram

A diagram is line art that your project draws itself, for a figure the graphics set does not have: a wiring plan, a timing chart, a state machine. Write it as an SVG file and place it with `::diagram name Alt text`. The name is the path of the file below a diagram folder, without `.svg`. The folder is `diagrams` below the source folder, and `diagrams` in `docs.config.json` lists other folders:

DOCS.CONFIG.JSON

```
{ "diagrams": ["diagrams", "figures"] }
```

`::diagram wiring/board-to-monitor` reads `docs-src/diagrams/wiring/board-to-monitor.svg`. The alt text is required, because a diagram is never decoration.



The diagram follows the design system's rules for line art, and `jupiter-docs check` tests them:

RULE	WHAT TO DO
Size	Give the `<svg>` a `viewBox` at most 720 wide, and keep the file under 100 KB.
Elements	Use `svg`, `g`, `defs`, `marker`, `title`, `desc`, `text`, `tspan`, `path`, `line`, `polyline`, `polygon`, `rect`, `circle` and `ellipse`.
Lines	Set `fill="none"`, `stroke="currentColor"` and one `stroke-width` between 1 and 3 on the `<svg>`. Shapes have no fills.

RULE	WHAT TO DO
Colour	Use `currentColor`, `none` or `var(--jp-NAME)`, where `NAME` is `text`, `text-muted`, `border`, `accent`, `ok`, `warn`, `danger` or `info`. Never a hex value, and no opacity.
Text	Do not set `font-family`. Diagram text is JetBrains Mono. A size of 13 or 14 units is readable in the PDF.
Nothing else	No gradients, filters, shadows, images, scripts, `style` attributes or links to other files.

An element outside the list, a `style` attribute, an event handler or a link is never published. In a project that docs.jupiter.au imports, other rule breaks are warnings, but one of these leaves the project off the site until it is fixed.

Text is filled and has no outline, whatever the `<svg>` sets. A marker does not inherit the colour of the line that uses it, so set `stroke` on the marker's own path.

On the site the SVG is inlined, so it follows the light and dark theme. For the Markdown output `jupiter-docs build` writes `docs/diagrams/<name>-light.svg` and `-dark.svg` and draws them with a `<picture>`, so commit them with the pages. The PDF build converts the light file with `rsvg-convert` and fails when that tool is missing, so a PDF never loses a diagram. A project's diagram folders are imported with its pages, so its diagrams appear on docs.jupiter.au.

03

Glossary

Terms used across the documentation of jupiter.au projects.

Graphic
A one-stroke SVG from the graphics set: an icon, a spot illustration, a scene or a rule.

Project
A repository with a docs.config.json and a docs-src folder. Its pages appear on this site under the project's id.

Token
A named design constant, such as a colour or a spacing step, defined once in jupiter.tokens.json and generated into every target.