




□□□□ >

Doom  [rspress](#)  Alauda          

□ □ □ □

- □□□□□□□□□□□□□□□□□□
- □□□□□□□□
- □□□□□

[illegible]



0000000000 MDX 00 ↗ 0

0000

0000000000 Doom 00

00

00

0000

000000

00

11

□□ `doom` □□□□

4/4

API ☐ ☐ ☐ ☐

□ □ □ □ □ □ □ □

Page 10 of 10

□ □ □ □ □

□ □ □ □ □ □ □ □

□ □ □ □ □ □ □ □ □

--	--	--	--	--	--	--	--

sites.yaml □□

□ □ □ □

□ □ □ □ □ □ □ □ □

Page 10

Algolia ☐☐☐☐

Sitemap ☐ ☐

11

□ □ □ □

3/3

11

11

Markdown

Callouts

Mermaid

MDX

00 MDX 000000000000000000

rspress 00

doom 00

00000000

000

0000000000000000

i18n.json

.ts/.tsx

.mdx

API 00

00 API

CRD

0000

00 openapi 00

000000

props

00

0000

000000

00

00000000000000000000 ACP 00

00000

00000

000000

00000000

📄📄

📄📄

📄📄📄

📄📄📄📄

📄📄📄📄📄

📄📄📄📄📄

📄📄📄

📄📄📄📄📄📄

📄📄📄

📄📄 PDF

📄📄📄

📄📄📄📄

📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

```
mkdir my-docs && cd my-docs
```

📄 `npm init -y` 📄📄📄📄📄📄📄📄📄📄📄📄 `npm`📄`yarn` 📄 `pnpm` 📄📄 `doom:`



```
npm install -D @alauda/doom typescript
```

📄📄📄📄📄📄📄📄:

```
# 创建 docs 目录
mkdir docs/en && echo '# Hello World' > docs/en/index.md
mkdir docs/zh && echo '# 你好' > docs/zh/index.md
```

在 `package.json` 中添加脚本:

```
{
  "scripts": {
    "dev": "doom dev",
    "build": "doom build",
    "new": "doom new",
    "serve": "doom serve",
    "translate": "doom translate",
    "export": "doom export"
  }
}
```

在 `doom.config.yml` 中添加:

```
title: My Docs
```

在 `tsconfig.json` 中添加:

```
{
  "compilerOptions": {
    "jsx": "react-jsx",
    "module": "NodeNext",
    "moduleResolution": "NodeNext",
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "resolveJsonModule": true,
    "skipLibCheck": true,
    "strict": true,
    "target": "ESNext",
  },
  "mdx": {
    "checkMdx": true,
  },
}
```

global.d.ts

```
/// <reference types="@alauda/doom/runtime" />
```

.mdx doom


```
doom -h
```

```
# output
```

```
Usage: doom [options] [command]
```

```
Doctor Doom making docs.
```

Options:

-V, --version	output the version number
-c, --config <config>	Specify the path to the config file
-v <version>	Specify the version of the documentation, c
-b, --base <base>	Override the base of the documentation
-p, --prefix <prefix>	Specify the prefix of the documentation bas
-f, --force [boolean]	Force to <ol style="list-style-type: none"> 1. fetch latest reference remotes or scaffo 2. translate ignore hash equality check and
-i, --ignore [boolean]	Ignore internal routes (default: false)
-d, --download [boolean]	Display download pdf link on nav bar (defau
-e, --export [boolean]	Run or build in exporting PDF mode, `apis/*
-I, --include <language...>	Include only the specific language(s),
-E, --exclude <language...>	Include all languages except the specific l
-o, --out-dir <path>	Override the `outDir` defined in the config
-r, --redirect <enum>	Whether to redirect to the locale closest t
-R, --edit-repo [boolean url]	Whether to enable or override the `editRepo
-a, --algolia [boolean alauda]	Whether to enable or use the alauda (docs.a
-S, --site-url	Whether to enable the siteUrl for sitemap g
-n, --no-open [boolean]	Do not open the browser after starting the
-h, --help	display help for command

Commands:

dev [options] [root]	Start the development server
build [root]	Build the documentation
preview serve [options] [root]	Preview the built documentation
new [template]	Generate scaffolding from templates
translate [options] [root]	Translate the documentation
export [options] [root]	Export the documentation as PDF, `apis/**`
lint [root]	Lint the documentation
help [command]	display help for command

000000

00 yarn dev 00000000000000000000

```
doom dev -h
```

```
# output
```

```
Usage: doom dev [options] [root]
```

```
Start the development server
```

```
Arguments:
```

```
root
```

```
Root directory of the documentation
```

```
Options:
```

```
-H, --host [host]
```

```
Dev server host name
```

```
-P, --port [port]
```

```
Dev server port number
```

```
-l, --lazy [boolean]
```

```
Whether to enable `lazyCompilation` which could i
```

```
-h, --help
```

```
display help for command
```

000000

00 yarn build 000000000000000000 dist 00000000

0000

00 yarn serve 000000000000000000 -b, -p 0000000000000000 -b, -p 0

0

00000000

00 yarn new 000000000000000000

0000

```
doom translate -h
```

```
# output
```

```
Usage: doom translate [options] [root]
```

Translate the documentation

Arguments:

root Root directory of the documentation

Options:

-s, --source <language> Document source language, one of en, zh, ru (default)

-t, --target <language> Document target language, one of en, zh, ru (default)

-g, --glob <path...> Glob patterns for source dirs/files

-C, --copy [boolean] Whether to copy relative assets to the target directory

-h, --help display help for command

- -g, --glob 指定源目录的glob模式，glob 模式指定了源目录的搜索范围。

1. yarn translate -g abc xyz 指定 <root>/<source>/abc , <root>/<source>/xyz 为源目录，<root>/<target>/abc , <root>/<target>/xyz 为目标目录。

2. yarn translate -g '*' 指定 <root>/<source> 为源目录。

- -C, --copy 指定是否将源目录中的相对资产复制到目标目录，false 表示不复制。

- 示例

1. /<source>/abc.jpg 复制到 <root>/public/<source>/abc.jpg 并复制到 <root>/public/<target>/abc.jpg 并复制到 <root>/<target>/abc.jpg

2. <root>/<source>/abc.mdx 复制到 ./assets/xyz.jpg 并复制到 <root>/<source>/assets/xyz.jpg 并复制到 <root>/<target>/assets/xyz.jpg 并复制到 <root>/<target>/assets/xyz.jpg

3. <root>/<source>/abc.mdx 复制到 ./assets/<source>/xyz.jpg 并复制到 <root>/<source>/assets/<source>/xyz.jpg 并复制到 <root>/<target>/assets/<target>/xyz.jpg 并复制到 ./assets/<target>/xyz.jpg

- 📄📄📄📄📄📄
 1. `/<source>/abc.jpg` 📄📄📄 `<root>/public/<target>/abc.jpg` 📄📄📄📄📄📄
📄📄📄📄 `/<target>/abc.jpg` 📄📄📄📄📄📄📄📄📄📄
 2. `<root>/<source>/abc.mdx` 📄📄 `./assets/<source>/xyz.jpg` 📄📄📄
`<root>/<target>/assets/<target>/xyz.jpg` 📄📄📄📄📄📄📄📄📄📄
`./assets/<target>/xyz.jpg` 📄📄📄📄
`../<source>/assets/<target>/xyz.jpg`

WARNING

📄📄📄📄📄 `-g '*'` 📄📄📄📄📄📄 `source` 📄 `target` 📄📄📄📄📄
`internalRoutes` 📄📄📄📄 `target` 📄📄📄📄📄

TIP

📄📄📄📄📄📄 `AZURE_OPENAI_API_KEY` 📄📄📄📄📄📄📄 `Leader` 📄

📄📄📄📄📄📄📄📄📄📄📄📄

```
i18n:
  title:
    en: DevOps Connectors
  additionalPrompts: '📄📄📄 Connectors 📄📄📄📄📄📄'
  disableAutoTranslation: false
title: DevOps 📄📄
```

📄📄📄📄📄📄📄📄📄

📄📄 **PDF**

WARNING

📄📄📄📄📄📄📄📄 `yarn build` 📄📄📄

```
doom export -h
```

```
# output
```

```
Usage: doom export [options] [root]
```

```
Export the documentation as PDF, `apis/**` and `*/apis/**` routes will be ign
```

```
Arguments:
```

```
  root                Root directory of the documentation
```

```
Options:
```

```
-H, --host [host]  Serve host name
-P, --port [port]  Serve port number (default: "4173")
-h, --help          display help for command
```

```
❯ yarn export 📄📄📄 PDF 📄📄📄📄📄📄📄 -b , -p 📄📄📄📄📄📄📄📄 -b , -p
❯
```

```
📄📄📄📄 playwright ↗📄📄📄📄📄 build-harbor.alauda.cn/frontend/playwright-
runner:doom 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄
```

```
.env.yarn
```

```
PLAYWRIGHT_DOWNLOAD_HOST="https://cdn.npmmirror.com/binaries/playwright"
```

```
📄📄📄📄
```

```
doom lint -h
```

```
# output
```

```
Usage: doom lint [options] [root]
```

```
Lint the documentation
```

```
Arguments:
```

```
  root          Root directory of the documentation
```

```
Options:
```

```
  -h, --help  display help for command
```



📄

📄 `doom` 📄📄

📄📄

📄📄

API 📄📄

📄📄📄📄

📄📄📄

📄📄📄

📄📄📄

📄📄📄📄

📄📄📄📄📄

📄📄📄📄

`sites.yaml` 📄

📄📄

📄📄📄📄📄

📄📄📄

Algolia 📄📄

Sitemap 📄

API 00

00 API

CRD

0000

00 openapi 00

000000

props

00

0000

000000

00

00000000000000000000 ACP 00

00000

00000

000000

00000000

11

Sitemap ☐ ☐

--	--	--	--

```
doom.config.yml  rspress  js/ts
```

```
□□□□ .js/.ts/.mjs/.mts/.cjs/.cts □□□□□□□□
```

```

js/ts @alauda/doom/config defineConfig

```

```
import { defineConfig } from '@alauda/doom/config'

export default defineConfig({})
```

□ □ □ □

- `lang` 指定する言語のコード (例: `en`)
指定しない場合は `null` または `undefined`
- `title` 記事のタイトル
- `logo` 記事のロゴ (例: `public`)
指定しない場合は `doom` または `alauda logo`
- `logoText` ロゴのテキスト (例: `logo`)
- `icon` アイコン (例: `favicon`)
- `base` ベース URL (例: `product-docs`)
- `outDir` 出力ディレクトリ (例: `dist/{base}/{version}`)
指定しない場合は `dist/{outDir}/{version}`

API ☐ ☐ ☐ ☐

api:

```
# CRD 配置 doom.config.* 文件 glob 模式 json/yaml 文件
```

crds:

```
- docs/shared/crds/*.yaml
```

```
# OpenAPI 配置 doom.config.* 文件 glob 模式 json/yaml 文件
```

openapis:

```
- docs/shared/openapis/*.json
```

```
# 配置 openapi 文件
```

```
# 配置 https://doom.alauda.cn/apis/references/CodeQuality.html#v1alpha1.CodeQ
```

references:

```
  v1alpha1.CodeQualityBranch: /apis/references/CodeQualityBranch#v1alpha1.C
```

```
# 配置 API 网关 gateway 文件
```

```
pathPrefix: /apis
```

配置 API 文件

配置 API 文件

```
# 配置 doom.config.* 文件 glob 模式 json/yaml 文件
```

permission:**functionresources:**

```
# `kubectl get functionresources`
```

```
- docs/shared/functionresources/*.yaml
```

roletemplates:

```
# `kubectl get roletemplates -l auth.cpaas.io/roletemplate.official=true`
```

```
- docs/shared/roletemplates/*.yaml
```

配置 API 文件

配置 API 文件

reference:

```

- repo: alauda-public/product-doc-guide # 
branch: # [string] 
publicBase: # [string] /images/xx.png do
sources:
  - name: anchor # 
    path: docs/index.mdx# # doom.conf
    ignoreHeading: # [boolean] true
    processors: 
      - type: ejTemplate
        data: # ejs `<%= data.xx %>` 
    frontmatterMode: merge # frontmatter ignore ign

```

frontmatterMode

- **ignore** frontmatter frontmatter
- **merge** frontmatter key
- **replace** frontmatter frontmatter
- **remove** frontmatter

releaseNotes:**queryTemplates:**

fixed: # ejs jql

unfixed:

release-notes.md

```
<!-- release-notes-for-bugs?template=fixed&project=DevOps -->
```

release-notes.mdx

```
{/* release-notes-for-bugs?template=fixed&project=DevOps */}
```

```
template=fixed&project=DevOps fixed queryTemplates
query project=DevOps ej ↗ fixed jira jql ↗
https://jira.alauda.cn/rest/api/2/search?jql=<jql> API
JIRA_USERNAME JIRA_PASSWORD
```

--	--	--	--	--

```
sidebar:
  collapsed: false # ☐ false
```

--	--	--	--	--	--	--	--

```
internalRoutes: # glob docs cli -i, --ignore`
- '*/internal/**'
```

--	--	--	--	--	--	--	--	--

```
onlyIncludeRoutes: # glob docs cli -i, --ignore`
  - '*/internal/**'
internalRoutes:
  - '*/internal/overview.mdx'
```



shiki:

theme: # optional, <https://shiki.style/themes>

langs: # optional, <https://shiki.style/languages>

transformers: # optional, only available in js/ts config, <https://shiki.style>

WARNING

plaintext

`sites.yaml` ☐ ☐

sites.yaml □□□
□□□□

```
- name: connectors # □□□□□□
```

```
base: /devops-connectors # □□□□□□□□
```

```
version: v1.1 # ExternalSite/ExternalSiteLink
```

```
displayName: #  name
```

en: DevOps Connectors

zh: DevOps □□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
repo: https://github.com/AlaudaDevops/connectors-operator # 000000000000 gi
```

image: devops/connectors-docs #

□ □ □ □

translate:

```
# ===== `sourceLang`, `targetLang`, `userPrompt` o `additionalPrompts` =====
#   `sourceLang` o `targetLang` o `` o ``
#       `userPrompt`
#       `additionalPrompts` o `frontmatter.i18n` o `additionalPrompts` o
#
systemPrompt: |
##
===== targetLang %>===== sourceLang %>o=<%=
##
-   sourceLang %>===== targetLang %>o=<%=
-   MDX MDX MDX jsx <Overview />
-
-   MDX MDX MDX MDX
    - <Overview /> o Overview
    - <Tab label="value"></Tab>label o key "value"
<%= terms %>
-
- {/* release-notes-for-bugs */}
- <!-- release-notes-for-bugs -->
-
- {/* reference-start */}
- {/* reference-end */}
- <!-- reference-start -->
- <!-- reference-end -->
- \< o \\{
- Markdown frontmatter, frontmatter.i18n
##
=====
1.   sourceLang %>===== targetLang %>=====
2.
   -   targetLang %>=====
   -
   -
3.   ===== targetLang %>=====
4.   ===== targetLang %>===== targetLang %>=====
=====
<%= userPrompt %>
```



```
<%= additionalPrompts %>
```

```
userPrompt: # 00000000 `systemPrompt` 00 `ejs` 000000
```

```
editRepoBaseUrl: alauda/doom/tree/main/docs # https://github.com/
```



```
lint:
```

```
cspellOptions: # cspell https://github.com/streetsidesoftware/csp
```

Algolia ☐☐☐☐

```
algolia: #   Algolia   `--a, --algolia`
```

appId: # Algolia  ID

apiKey: # Algolia API Key

indexName: # Algolia □□□□

public/robots.txt Algolia

INFO

rspress Algolia

@alauda/doom/theme

```
export * from '@alauda/doom/theme'
```

Sitemap 📄

```
siteUrl: https://docs.alauda.cn # 📄 URL📄 sitemap📄 ` -S, --site-url
```

11

11

--	--	--	--

```

index
index.md index.mdx

```

```
├─ index.md
├─ start.mdx
└─ usage
    ├─ index.mdx
    └─ convention.md
```

--	--	--	--	--	--

1. `public` □□□□□□□□□□□□□□□□
2. `public/_remotes` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
`*/public/_remotes` □□ `.gitignore` □□□□□□□□□□
3. `shared` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

🐱🐱

🐱🐱🐱🐱🐱🐱 `frontmatter` 🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱

```
---
title: 🐱
description: 🐱
author: 🐱
category: 🐱
---
```

🐱🐱🐱🐱🐱🐱 `MDX` 🐱 `.mdx` 🐱🐱🐱🐱🐱 `frontmatter` 🐱🐱🐱🐱🐱🐱

🐱🐱

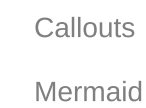
🐱 `index.md` 🐱 `index.mdx` 🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱 `frontmatter` 🐱
`weight` 🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱 `weight` 🐱🐱🐱🐱🐱🐱🐱

```
---
weight: 1
---
```

WARNING
🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱🐱

🐱🐱

gfm ↗ Doom Markdown



□ □ □ □ □ □

1. 在代码块中，使用 `;`, `%`, `#`, `//`, `/** */`, `--` 和 `<!-- -->` 来插入特殊字符。
2. 在代码块中，使用 `[!code callout]` 来插入 callout。
3. 在代码块中，使用 `:::callouts` 来插入 callouts。在代码块中，使用 `<div class="doom-callouts">` 来插入 callouts。在代码块中，使用 `<Callouts>` 来插入 callouts。

```
```sh
```

```
Memory overhead per virtual machine ≈ (1.002 × requested memory) \
 + 218 MiB \
 + 8 MiB × (number of vCPUs) \
 + 16 MiB × (number of graphics devices) \
 + (additional memory overhead)
```

```
```
```

```
:::callouts
```

1. Required for the processes that run in the `virt-launcher` pod.
2. Number of virtual CPUs requested by the virtual machine.
3. Number of virtual graphics cards requested by the virtual machine.
4. Additional memory overhead:
 - If your environment includes a Single Root I/O Virtualization (SR-IOV) network device or a Graphics Processing Unit (GPU), allocate 1 GiB additional memory overhead for each device.
 - If Secure Encrypted Virtualization (SEV) is enabled, add 256 MiB.
 - If Trusted Platform Module (TPM) is enabled, add 53 MiB.

```
:::
```

```
Memory overhead per virtual machine ≈ (1.002 × requested memory) \
    + 218 MiB \
    + 8 MiB × (number of vCPUs) \
    + 16 MiB × (number of graphics devices) \
    + (additional memory overhead)
```

- ① Required for the processes that run in the `virt-launcher` pod.
- ② Number of virtual CPUs requested by the virtual machine.
- ③ Number of virtual graphics cards requested by the virtual machine.
- ④ Additional memory overhead:
 - If your environment includes a Single Root I/O Virtualization (SR-IOV) network device or a Graphics Processing Unit (GPU), allocate 1 GiB additional memory overhead for each device.
 - If Secure Encrypted Virtualization (SEV) is enabled, add 256 MiB.
 - If Trusted Platform Module (TPM) is enabled, add 53 MiB.

Mermaid ↗

□□□□□

```
```mermaid
graph TD;
 A-->B;
 A-->C;
 B-->D;
 C-->D;
```
```



□□ [Markdown Preview Mermaid ↗](#) □□□ VSCode □□□□□□

MDX

[MDX](#) [Markdown](#) [Markdown](#) [JSX](#) [rspress MDX](#)

rspress

doom

Overview

Directive

ExternalSite

ExternalSiteLink

AcpApis0overview ExternalApis0overview

Term

props

TermsTable

props

JsonViewer

rspress

rspress [.mdx](#)

- Badge

- [Card](#)
- [LinkCard](#)
- [PackageManagerTabs](#)
- [Steps](#)
- [Tab/Tabs](#)
- [Toc](#)

安装 @rspress/core/theme 包

```
preview.mdx

import { SourceCode } from '@rspress/core/theme'

<SourceCode href="/" />
```

doom 文档

doom 是一个基于 Vite 的静态站点生成器，支持 MDX 和 Markdown 文件，并提供了丰富的主题和布局。

Overview

doom 支持以下功能：

Directive

doom 提供了 [Directive](#) 功能，用于在 MDX 文件中插入自定义内容。

```
- 在 `doc/en` 目录下创建 `doc/zh` 目录，用于存放中文文档。

<Directive type="danger" title="警告">
  这是一个危险的操作，可能会导致数据丢失。
  `doc/zh` 目录下创建 `doc/zh` 目录。
</Directive>
```

- 在 `(doc/en)` 中指定语言 `doc/zh` 以指定文档语言。默认语言为 `doc/en`。

注意

在 `doc/zh` 中指定语言。

ExternalSite

在文档中

```
<ExternalSite name="connectors" />
```

Note

在 `DevOps` 文档中指定语言。在 `DevOps` 文档中指定语言。
[DevOps 文档](#)

ExternalSiteLink

在文档中

```
<ExternalSiteLink name="connectors" href="link.mdx#hash" children="Content" />
```

[Content](#)

TIP

在 `mdx` 中指定 `<ExternalSiteLink name="connectors" href="link" children="Content" />`。
在文档中

```
<ExternalSiteLink name="connectors" href="link">
  Content {/*   `p`   */}
</ExternalSiteLink>
```

p children

AcpApisOverview ExternalApisOverview

API

```
<AcpApisOverview />
{/* same as following */}
<ExternalApisOverview name="acp" />

<ExternalApisOverview name="connectors" />
```

Note

ACP APIs [ACP APIs](#)

Note

DevOps APIs [DevOps APIs](#)

Term

```
<Term name="company" textCase="capitalize" />
<Term name="product" textCase="lower" />
<Term name="productShort" textCase="upper" />
```

ACP

props

- `name` :
- `textCase` : `lower` , `upper` , `capitalize`

TermsTable

```
<TermsTable />
```

company		-	Alauda	-	
product		-	Alauda Container Platform	-	
productShort	ACP	-	ACP	-	

props

- `terms` : `NormalizedTermItem[]`

JsonViewer

```
<JsonViewer value={{ key: 'value' }} />
```

```
yaml    json
key: value
```

□□□□□□□□

□□□□□□□□□□□□□□□□□□ `shared` □□□□□□□□□□□□□□□□□□□□

```
import CommonContent from './shared/CommonContent.mdx'

<CommonContent />
```

□□□□□□□□ `runtime` ↗ □□□ API□□□□□□ `.jsx/.tsx` □□□□□□□□ `.mdx` □□□□□□□□

```
// shared/CommonContent.tsx
export const CommonContent = () => {
  const { page } = usePageData()
  return <div>{page.title}</div>
}

// showcase/content.mdx
import { CommonContent } from './shared/CommonContent'
<CommonContent />
```

WARNING

□□□□□□ `.mdx` □□□□□□□□□□ `props` □□□□□□ issue ↗ □□□□□□□□ `props` □□□□□□□□
`.jsx/.tsx` □□□□□□□□

📄📄📄

`alauda` 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄 `en / zh` 📄📄📄📄📄📄📄📄📄📄📄📄
📄 `public` 📄📄📄📄 `en / zh` 📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄

📄📄

- `i18n.json`
- `.ts/.tsx`
- `.mdx`

`i18n.json`

📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄📄 `docs` 📄📄📄📄 `i18n.json` 📄📄
📄📄📄📄📄📄 `useI18n` 📄📄📄📄📄📄📄📄📄📄📄📄

docs/i18n.json

```
{
  "title": {
    "zh": "标题",
    "en": "Title"
  },
  "description": {
    "zh": "描述",
    "en": "description"
  }
}
```

.ts/.tsx

```
import { useI18n } from '@rspress/runtime'

export const CommonContent = () => {
  const t = useI18n()
  return <h1>{t('title')}</h1>
}
```

.mdx

```
import { useI18n } from '@rspress/runtime'

# {useI18n()('title')}

{useI18n()('description')}
```


API 文档

API 文档 API 文档 CRD (Custom Resource Definition) 文档

```
├─ apis
│   ├── advanced-apis # API
│   ├── crds # CRDs
│   └── references # 文档
```

文档

API

props

CRD

props

文档

props

openapi 文档

API

advanced-apis/codeQualityTaskSummary.mdx

```
# CodeQualityTaskSummary

<OpenAPIPath path="/plugins/v1alpha1/template/codeQuality/task/{task-id}/summ
```

CodeQualityTaskSummary

props

- path : OpenAPI schema paths
- pathPrefix : api.pathPrefix
- openapiPath : openapi

CRD

crds/ArtifactCleanupRun.mdx

```
# ArtifactCleanupRun

<K8sCrd name="artifactcleanupruns.artifacts.katanomi.dev" />
```

ArtifactCleanupRun

props

- name : CRD metadata.name
- crdPath : openapi CRD

references/CodeQuality.mdx

CodeQuality

```
<OpenAPIRef schema="v1alpha1.CodeQuality" />
```

関数 `CodeQuality`

props

- `schema` : OpenAPI schema `definitions` (v2) or `component/schemas` (v3) から取得
- `openapiPath` : 指定した `openapi` のパス

関数 `openapi`

関数 `OpenAPIPath` は `OpenAPIRef` から取得した `openapi` のスキーマからパスを抽出する関数。
関数 `openapi` は `openapiPath` からスキーマを取得する。

```
<OpenAPIPath  
  path="/plugins/v1alpha1/template/codeQuality/task/{task-id}/summary"  
  openapiPath="shared/openapis/katanomi.json"  
>
```

权限管理

```
<K8sPermissionTable functions={['devops-testplans', 'devops-testmodules']} />
```

权限

props

权限

props

- functions: string[] - 权限资源 FunctionResource 列表

权限

权限	权限	权限 名称	权限 名称	权限 名称	权限 名称	权限 名称	权限 名称
权限 devops- testplans	权限	✓	✓	✓	✓	✓	✗
	权限	✓	✗	✓	✓	✓	✗
	权限	✓	✗	✓	✓	✓	✗
	权限	✓	✗	✓	✓	✓	✗

00	00	000 00	000 000	000 00	0000 000	00 00	000 00
0000 devops- testmodules	00	✓	✓	✓	✓	✓	×
	00	✓	×	✓	✓	✓	×
	00	✓	×	✓	✓	✓	×
	00	✓	×	✓	✓	✓	×

--	--	--	--

☐ Markdown ☐☐☐☐

```
<!-- reference-start#name -->
```

```
<!-- reference-end -->
```

☐ MDX ☐☐☐☐☐

```
{/* reference-start#name */}
```

```
{/* reference-end */}
```

name

```
□□□□□□ <root>/public/_remotes/<name> □□□□
```

```
□□□□□ <!-- reference-start#ref --> □□□□□
```

□ □ □ □ □ □

frontmatterMode

--	--	--	--	--	--

reference:

```

- repo: alauda-public/product-doc-guide # 
branch: # [string] 
publicBase: # [string] /images/xx.png do
sources:
  - name: anchor # 
    path: docs/index.mdx# # doom.conf
    ignoreHeading: # [boolean] true
    processors: # 
      - type: ejTemplate
        data: # ejs `<%= data.xx %>` 
    frontmatterMode: merge # frontmatter ignore ign

```

frontmatterMode

- **ignore** frontmatter frontmatter
- **merge** frontmatter key
- **replace** frontmatter frontmatter
- **remove** frontmatter

11

--	--	--	--	--	--	--	--

--	--	--	--	--

[illegible]

```
doom build # ████████
doom serve # ██████████
```

--	--	--	--	--

```
doom build dist -v
```



```
# ===== release-4.0 == 4.0 ==
doom build -v 4.0 # == 4.0 ===== dist/4.0===== {base}/4.0
doom build -v master # == master ===== dist/master===== {base}/master
doom build -v {other} # ===== dist/{other}===== {base}/{other}

# unversioned = unversioned-x.y =====
doom build -v unversioned # ===== dist/unversioned===== {base}
doom build -v unversioned-4.0 # ===== 4.0 ===== dist/unversio
```

□□□□□□

```
|— console-platform
|   |— 4.0
|   |— 4.1
|   |— index.html
|   |— overrides.yaml
|   └— versions.yaml
|— console-devops-docs
|   |— 4.0
|   |— 4.1
|   |— index.html
|   |— overrides.yaml
|   └— versions.yaml
|— console-tekton-docs
|   |— 1.0
|   |— 1.1
|   |— index.html
|   |— overrides.yaml
|   └— versions.yaml
```

index.html



```
#   □□□□□□□□□□□□□□□□□□  
title:  
    en: Doom - Alauda  
    zh: Doom - □□□  
  
logoText:  
    en: Doom - Alauda  
    zh: Doom - □□□
```

versions.yaml

- '4.1'
- '4.0'