□□□□

Doom 是基于 [rspress](#) 开发的，由的是 Alauda □□。

□□

□□□□

支持 Markdown 语法以及 MDX

□□□□

□□□□

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

支持 Markdown 语法以及 MDX

MDX □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Markdown □□□□□□□□□ Markdown □□□□□□□ React □□□。

```
// docs/index.mdx
import { CustomComponent } from './custom';

# Hello World

<CustomComponent />
```

□□□□□□□□□□ MDX□ □□□

□□□□

□□□□□□□□□□□ Doom □□

□□

> □□
>
> □□□□
>
> □□□□□

□□

□□

使用 `doom` 构建站点

□□□□

□□□□

API □□□□

□□□□□□□□

□□□□□□

□□□□□□

□□□□□

□□□□□□

□□□□□□□

`sites.yaml` □□

□□□□

□□

关于"□□□□□□"□□□

□□□□

□□□

□□

□□

Markdown

Callouts

Mermaid

MDX

关于 MDX 的用法以及我们在此基础上的增强：

rspress 基础

doom 增强

自定义组件等

国际化

我们支持三种国际化的方式：

`i18n.json`

`.ts/.tsx`

`.mdx`

API 文档

关于 API

CRD

字段说明

以及 openapi 等。

前端组件库文档

`props`

示例

更多功能

更多实用功能

□□

□□□□□□□□□□□□□□□□□□□□□ ACP □□

□□□□□

□□□□

□□□ ACP

□□

□□

□□□□
□□□□□
□□□□□□
□□□□□□
□□□□
□□□□□□□
□□□□
□□ PDF

□□□□

□□□□□□□□□□□□□□□□□□□□□□

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

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

npm   **yarn**   **pnpm**   **bun**
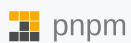
```
npm install -D @alauda/doom typescript
```

□□□□□□□□□□□□□□:

```
# □□ docs □□□□□□□□□□□□
mkdir docs/en && echo '# Hello World' > docs/en/index.md
```

```
mkdir docs/zh && echo '# □□□□' > docs/zh/index.md
```

□ `package.json` □□□□□□□□□:

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

□□□□□□□□□□□ `doom.config.yml`:

```yml
title: My Docs
```

□□□□ `tsconfig.json` □□□□□□□□

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

你可以在 `global.d.ts` 文件中添加以下内容：

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

以便在你的 `.mdx` 文件中获得更好的 doom 运行时类型提示支持。

命令行工具

```
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
                              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
  -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/**`
  help [command]                 display help for command
```

启动开发服务

通过 `yarn dev` 可以启动一个本地的开发服务来预览文档。

```
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
```

构建静态产物

通过 `yarn build` 可以将文档构建为静态产物，默认会输出到 `dist` 目录，用于后续部署。

预览产物

通过 `yarn serve` 可以将构建出来的静态产物跑起来，通过 `-b`, `-p` 参数进行配置，其含义与上文的 `-b`, `-p` 一致。

生成脚手架模版

通过 `yarn new` 可以根据内置的一些模版来生成脚手架。

翻译文档

```
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 (defau
  -t, --target <language>  Document target language, one of en, zh, ru (defau
  -g, --glob <path...>     Glob patterns for source dirs/files
  -C, --copy [boolean]     Wether to copy relative assets to the target direc
  -h, --help               display help for command
```

- `-g, --glob` 用于指定翻译的源文件或文件夹，通过 `glob` 匹配。如果不指定，则翻译整个文档目录。注意相对路径是相对于源目录的。

  1. `yarn translate -g abc xyz` 会翻译 `<root>/<source>/abc`, `<root>/<source>/xyz` 目录下的文件，并输出到 `<root>/<target>/abc`, `<root>/<target>/xyz` 目录下

  2. `yarn translate -g '*'` 会翻译 `<root>/<source>` 目录下的所有文件

- `-C, --copy` 用于指定是否复制相对引用的资源文件到目标目录。默认为 `false`。假设要翻译的文件中引用了以下资源文件：

  - 对于绝对路径

    1. `/<source>/abc.jpg` 会被解析为 `<root>/public/<source>/abc.jpg` 并复制到 `<root>/public/<target>/abc.jpg`，同时原始引用会被替换为 `/<target>/abc.jpg`

    2. `<root>/<source>/abc.mdx` 中引用的 `./assets/xyz.jpg` 会被解析为 `<root>/<source>/assets/xyz.jpg` 并复制到 `<root>/<target>/assets/xyz.jpg`，同时原始引用保持不变

    3. `<root>/<source>/abc.mdx` 中引用的 `./assets/<source>/xyz.jpg` 会被解析为 `<root>/<source>/assets/<source>/xyz.jpg` 并

`<root>/<target>/assets/<target>/xyz.jpg`，在原文中应引用为 `./assets/<target>/xyz.jpg`

- 当翻译图片引用时：

1. `/<source>/abc.jpg` 应转换为 `<root>/public/<target>/abc.jpg`，这样才能在翻译后的文档中引用为 `/<target>/abc.jpg`，否则会因为路径不一致而失效。

2. `<root>/<source>/abc.mdx` 中引用的 `./assets/<source>/xyz.jpg` 应被转换为 `<root>/<target>/assets/<target>/xyz.jpg`，这样才能在翻译后的文档中引用为 `./assets/<target>/xyz.jpg`，否则会失效为 `../<source>/assets/<target>/xyz.jpg`

> **WARNING**
>
> 由于翻译时需要 `-g '*'` 参数来匹配所有文件，因此 `source` 和 `target` 需要保持一致，否则 `internalRoutes` 的翻译会导致 `target` 的文件无法被匹配。

> **TIP**
>
> 如果需要配置自定义的 `AZURE_OPENAI_API_KEY`，请在翻译前联系文档库的 Leader 处理。

对于某些页面，你还可以进行如下配置：

```
i18n:
  title:
    en: DevOps Connectors
  additionalPrompts: '这是一个 Connectors 相关的页面标题'
  disableAutoTranslation: false
title: DevOps 连接器
```

导出 PDF

> **WARNING**
>
> 在导出前请务必确保已执行 `yarn build` 构建项目

```
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
```

其中 `yarn export` 相关命令的 PDF 导出参数不支持配置 `-b`, `-p` 参数，因为内部写死了相关 `-b`, `-p` 参数

该命令依赖于 `playwright`，我们提供了内置 `build-harbor.alauda.cn/frontend/playwright-runner:doom` 镜像，已经安装了相关依赖，可直接使用。 如果需要自行安装，可参考如下命令：

.env.yarn

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

□□

□□

□□ `doom` □□□□

□□□□

□□□□

API □□□□

□□□□□□□□

□□□□□□

□□□□□□

□□□□□

□□□□□

□□□□□□□□

`sites.yaml` □□

□□□□

□□

□□"□□□□□□□"□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□

□□□

□□

□□

Markdown

Callouts

Mermaid

MDX

□□ MDX □□□□□□□□□□□□□□□□□□□□

rspress □□

doom □□

□□□□□□□

□□□

□□□□□□□□□□□□□□□□

`i18n.json`

`.ts/.tsx`

`.mdx`

API □□

□□ API

CRD

□□□□

□□ openapi □□

□□□□□□□

`props`

□□

□□□□

□□□□□□

目录

在本章中，你将了解什么是任务，以及如何通过 ACP 协议

与别的智能体

与别的智能

什么是 ACP

□□

□□

□□□□

□□□□

API □□□□

□□□□□□□□

□□□□□□□

`frontmatterMode`

□□□□□□

□□□□□

□□□□□□

□□□□□□□□

`sites.yaml` □□

□□□□

□□□□

□□□□□□□□□□□□□□□□ `yaml` □□□□□□□□□□ `doom.config.yaml` □ `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({})
```

配置项

- `lang` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `en` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `null` □ `undefined`

- `title` □□□□□□□□□□□□□□□□□□

- `logo` □□□□□□□ logo□□□□□□□□□□□□□□□□□□□ `public` □□□□□□□□□□□□□□□□□□□□□□□□□ `doom` □□□□ alauda logo

- `logoText` □□□□□□□□□□□□□ logo □

- `icon` □□□ favicon□□□□ `logo`

- `base` □□□□□□□□□□□□□□□□□ `product-docs` □□□□ `/`

- `outDir` □□□□□□□□□□ `dist/{base}/{version}` □□□□□□□□□□□□ `dist/{outDir}/{version}` □□□ `version` □□□□□□□□□□□□

API 配置项

```
api:
  # CRD □□□□□□□□□ doom.config.* □□□□□□ glob □□□json/yaml □□
  crds:
    - docs/shared/crds/*.yaml
  # OpenAPI □□□□□□□□□ doom.config.* □□□□□□ glob □□□json/yaml □□
  openapis:
    - docs/shared/openapis/*.json
  # □□ openapi □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
  # □□ https://doom.alauda.cn/apis/references/CodeQuality.html#v1alpha1.CodeQ
  references:
    v1alpha1.CodeQualityBranch: /apis/references/CodeQualityBranch#v1alpha1.C
  # □□□API □□□□□□□□□□□□□□ gateway □□□□□□□□□□□□
  pathPrefix: /apis
```

□□□□□□□ [API □□](#)

□□□□□□□□

```yaml
# □□□□□□□□□□ doom.config.* □□□□□□□ glob □□□json/yaml □□
permission:
  functionresources:
    # `kubectl get functionresources`
    - docs/shared/functionresources/*.yaml
  roletemplates:
    # `kubectl get roletemplates -l auth.cpaas.io/roletemplate.official=true`
    - docs/shared/roletemplates/*.yaml
```

□□□□□□[□□□□□□](#)

□□□□□□

```yaml
reference:
  - repo: alauda-public/product-doc-guide # □□□□□□□□□□□□□□□□□□□□□□□□□□□□□
    branch: # [string] □□□□□□□□□□□
    publicBase: # [string] □□□□□□□□□□□□□□□ /images/xx.png □□□□□□□□□□□□□□ do
    sources:
      - name: anchor # □□□□□□□□□□□□□□□□□□□□
        path: docs/index.mdx#□□ # □□□□□□□□□□□□□□□□□□□□□□□□□□ doom.confi
        ignoreHeading: # [boolean] □□□□□□□□□□□ true□□□□□□□□□□□□□□□□□
        processors: # □□□□□□□□□□□□
          - type: ejsTemplate
            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` □□□□ `ejs` □□□□□□□□ `fixed` □□□□□□□□ jira `jql` □□ `https://jira.alauda.cn/rest/api/2/search?jql=<jql>` □□□□ API □□□□□□□□□ `JIRA_USERNAME` □ `JIRA_PASSWORD` □□□□□□□□□□

□□□□□

```
sidebar:
```

```yaml
    collapsed: false # □□□□□□□□□□□□□□□□□□□□□□□□□□ false
```

□□□□□□

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

□□□□□□□□

```yaml
shiki:
  theme: # optional, https://shiki.style/themes
  langs: # optional, https://shiki.style/languages
  transformers: # optional, only available in js/ts config, https://shiki.sty
```

> **WARNING**
>
> □□□□□□□□□□□□□□□□□□□□□□□□ `plaintext` □□

`sites.yaml` □□

`sites.yaml` □□
□□□□

```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 # □□□□□□□□□□□ gi
  image: devops/connectors-docs # □□□□□□□□□□□□□□□
```

□□□□

```
translate:
  # □□□□□□ejs □□□□□□□□□ `sourceLang`, `targetLang`, `userPrompt` □ `additiona
  # □□ `sourceLang` □ `targetLang` □ `□□` □ `□□` □□□□□□
  #     `userPrompt` □□□□□□□□□□□□□□
  #     `additionalPrompts` □□□ `frontmatter.i18n` □□ `additionalPrompts` □□□
  # □□□□□□□□□□□□□□□□□□
  systemPrompt: |
## □□
□□□□□□□□□□□□□□□□□□□□□□<%= targetLang %>□□□□□□□□□□□□□□□□<%= sourceLang %>□□□<%

## □□
- □□□□□□□□□□□□□<%= sourceLang %>□□□□□□□□□□□□□□□□□□□□□<%= targetLang %>□□□□□
- □□□□□ MDX □□□□□□□□□□□□□ MDX □□□□□□□□□□ jsx □□□□□□ <Overview />□□□□□□□□□□
- □□□□□□□□□□□□□□□
- MDX □□□□□□□□□□□□□MDX □□□□□□□□□□□□□□ MDX □□□□□□□□□□□
  - <Overview /> □□ Overview □□□□□□□□□□
  - <Tab label="value">□□□□□□□□</Tab>□label □ key □□□□□"value" □□□□□□□□
<%= terms %>
- □□□□□□□□□□□□□□□□□□□□□□□
  - {/* release-notes-for-bugs */}
  - <!-- release-notes-for-bugs -->
- □□□□□□□□□□□□□□□
  - {/* reference-start */}
  - {/* reference-end */}
  - <!-- reference-start -->
  - <!-- reference-end -->
- □□□□□□□□□□□□ \\< □ \\{ □□□□□□□□□□□□
- □□□□□□□□□□□ Markdown □□□□ frontmatter, □□□□□□□□□□□□ frontmatter.ii8n □□□□
```

```
## □□

□□□□□□□□□□
1. □□<%= sourceLang %>□□□□□<%= targetLang %>□□□□□□□□□□□□□□
2. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
 - □□□<%= targetLang %>□□□□□□□□□□□□□
 - □□□□□□□□□□□□□□□□□□□□□□□□□□□
 - □□□□□□□□□□□□□□□□□□□□□□□□
3. □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□<%= targetLang %>□□□□□□□
4. □□□□□□□□□□<%= targetLang %>□□□□□□□□□□□□□□□□□<%= targetLang %>□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

<%= userPrompt %>

<%= additionalPrompts %>
  userPrompt: # □□□□□□□□□ `systemPrompt` □□ `ejs` □□□□□□
```

□□

□□

□□□□

□□□

□□

□□

□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□ `index` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `index.md` □ `index.mdx` □□□

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

□□□□□□□

1. `public` □□□□□□□□□□□□□□□□□□□□□

2. `public/_remotes` □□ `*/public/_remotes` □□ `.gitignore` □□□□□□□□□□

3. `shared` □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□

你可以在文档中添加 `frontmatter` 来为文档提供标题、描述、作者等元数据信息。

```
---
title: □□
description: □□
author: □□
category: □□
---
```

你也可以在使用 [MDX](#) 格式 `.mdx` 的文档中添加 `frontmatter` 来提供元数据信息。

□□

在 `index.md` 或 `index.mdx` 这样的索引文件中，你可以通过在文档的 `frontmatter` 中添加 `weight` 字段来控制文档的排序。值越小的 `weight` 字段的文档排序越靠前。

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

> **WARNING**
>
> 如果两篇文档的权重值相同，它们将按照文件名的字母顺序进行排序。

□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ `index.mdx` □□□□□□ `Overview` □□□

```
# □□


<Overview />
```

□□□□□□□□□□□

Markdown

􀀀􀀀􀀀􀀀 [gfm](#) 􀀀􀀀􀀀􀀀Doom 􀀀􀀀􀀀􀀀􀀀􀀀􀀀 Markdown 􀀀􀀀􀀀􀀀

􀀀􀀀

Callouts

Mermaid

Callouts

􀀀􀀀􀀀􀀀􀀀

> **NOTE**
>
> 1. 􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀 `;` , `%` , `#` , `//` , `/** */` , `--` 􀀀 `<!-- -->` 􀀀
>
> 2. 􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀 `[\!code callout]` 􀀀􀀀􀀀􀀀
>
> 3. 􀀀􀀀􀀀 `:::callouts` 􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀􀀀 `<div class="doom-callouts">` 􀀀 `<Callouts>` 􀀀􀀀􀀀􀀀

```sh
Memory overhead per virtual machine ≈ (1.002 × requested memory) \
             + 218 MiB \  ①
             + 8 MiB × (number of vCPUs) \  ②
             + 16 MiB × (number of graphics devices) \  ③
             + (additional memory overhead)  ④
```

:::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) n
   - 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 \      1
            + 8 MiB × (number of vCPUs) \      2
            + 16 MiB × (number of graphics devices) \      3
            + (additional memory overhead)      4
```

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.

□□□□□□□□□□□ [Shiki Transformers]□

Mermaid

□□□□□□

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


使用 Markdown Preview Mermaid 插件的 VSCode 页面渲染效果

MDX
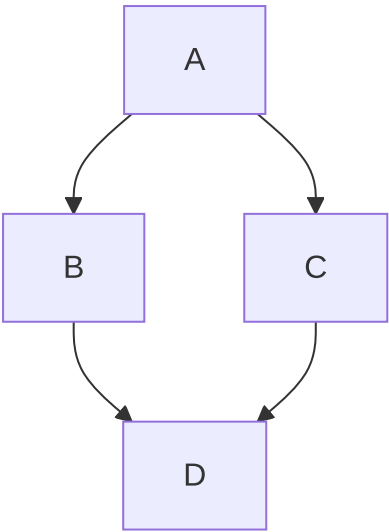
MDX □□□ Markdown □□□□□□□□□□□ Markdown □□□ JSX □□□□□□□□□□□□□ rspress MDX□

□□

rspress □□

doom □□

> `Overview`
>
> `Directive`
>
> `ExternalSite`
>
> `ExternalSiteLink`
>
> `AcpApisOverview` □ `ExternalApisOverview`
>
> Term
>
> > `props`
> >
> > `TermsTable`
> >
> > `props`
> >
> > `JsonViewer`
>
> □□□□□□□

---

rspress □□

`rspress` □□□□□□□□□□□□□□□□□□□□□□□□□ `.mdx` □□□□□□□□□□□□□□□□□□

- `Badge`
- `Card`
- `LinkCard`

- `PackageManagerTabs`

- `Steps`

- `Tab/Tabs`

- `Toc`

□□□□□□□□□□□□ `@rspress/core/theme` □□□□□□□□□

```
preview.mdx

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

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

doom □□

`doom` □□

Overview

□□□□□□□□□□□□□□□

Directive

□□□□□□□□□□□□□□□□□□□□□□□□□□□ `Directive` □□□□

```
- □□□□□(`doc/en`)□□□□□□□□ `doc/zh` □□□□□□□□□□□□□□□□□□□□□□□□□□□□

  <Directive type="danger" title="□□">
    □□□□□□□□□□□□□□□□□□□□□□□□□□□□
    `doc/zh` □□□□□□□□□□□□
  </Directive>
```

- 英文文档（ `doc/en` ）改动后，需要将对应 `doc/zh` 下的中文文档一并进行更新，以确保中英文文档内容的一致性。

> **警告**
>
> 在未与团队达成一致的情况下，请勿随意更改文档结构或文件名，否则会导致 `doc/zh` 下的中文文档无法正确对应英文文档。

ExternalSite

外部站点组件。

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

> **Note**
>
> 用于 DevOps 等外部站点的链接引用，确保链接到正确的 DevOps 站点页面。更多信息请参阅[DevOps 文档↗](#)。

ExternalSiteLink

外部站点链接组件。

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

[Content↗](#)

> **TIP**
>
> 在 mdx 中 `<ExternalSiteLink name="connectors" href="link" children="Content" />` 等价于以下写法：
>
> ```
> <ExternalSiteLink name="connectors" href="link">
>   Content {/* 这里是 `p` 标签 */}
> ```

```
    </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

```yaml
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` □□□□□□□

国际化

`alauda` 文档框架提供了完整的国际化（多语言）支持。默认支持 `en` / `zh` 两种语言，你可以根据需要扩展更多语言。在 `public` 目录下的 `en` / `zh` 目录中，你可以放置对应语言的静态资源文件，系统会根据当前语言自动加载。

分类

i18n.json

.ts/.tsx

.mdx

---

i18n.json

用于存放无法在其他文件中维护的多语言文案，通常以模块为单位，放置于 `docs` 目录下的 `i18n.json` 文件中。你可以通过 `useI18n` 钩子函数在组件中使用这些文案。

```
docs/i18n.json

{
  "title": {
    "zh": "□□",
    "en": "Title"
  },
  "description": {
    "zh": "□□",
    "en": "description"
  }
}
```

.ts/.tsx

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

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

.mdx

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

# {useI18n()('title')}

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

API □□

□□□□□□□□□□□□□ API □□□□ API □ CRD (Custom Resource Definition) □□□□□□□□□□□□□□□□□

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

□□

□□ API

`props`

CRD

`props`

□□□□

`props`

□□ openapi □□

□□ API

```
advanced-apis/codeQualityTaskSummary.mdx

# CodeQualityTaskSummary

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

查看 [CodeQualityTaskSummary]。

props

- `path` : OpenAPI schema `paths` 中的路径
- `pathPrefix` : 请求路径的前缀，默认使用 `api.pathPrefix`
- `openapiPath` : 同全局 [openapi 配置]

CRD

crds/ArtifactCleanupRun.mdx

```
# ArtifactCleanupRun

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

查看 [ArtifactCleanupRun]。

props

- `name` : CRD `metadata.name`
- `crdPath` : 同全局 [openapi 配置]，但用于单独指定 CRD 路径

模板示例

references/CodeQuality.mdx

```
# CodeQuality

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

查看 CodeQuality□

props

- `schema` : OpenAPI schema `definitions` (v2) or `component/schemas` (v3) 中的某项

- `openapiPath` : 可选,引用 openapi 文档

□□ openapi 文档

使用 `OpenAPIPath` 和 `OpenAPIRef` 组件时,一般需要引用 openapi 文档,当定义字段为全局引用时,需引用 openapi 文档,可以通过 `openapiPath` 属性来设置

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

□□□□□

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

□□

`props`
□□

props

- `functions` : `string[]` - □□□□□□□□ `FunctionResource` □□□□□□

□□

□□	□□	□□□□□	□□□□□□	□□□□□	□□□□□□□	□□□□	□□□□□
□□□□ `devops-testplans`	□□	✓	✓	✓	✓	✓	✕
	□□	✓	✕	✓	✓	✓	✕
	□□	✓	✕	✓	✓	✓	✕
	□□	✓	✕	✓	✓	✓	✕
□□□□ `devops-`	□□	✓	✓	✓	✓	✓	✕

□□	□□	□□□□□	□□□□□□	□□□□□	□□□□□□□	□□□□	□□□□□
`testmodules`	□□	✓	✗	✓	✓	✓	✗
	□□	✓	✗	✓	✓	✓	✗
	□□	✓	✗	✓	✓	✓	✗

□□□□

□ 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.confi
        ignoreHeading: # [boolean] □□□□□□□□□□□□□ true□□□□□□□□□□□□□□□□□□
        processors: # □□□□□□□□□□□□□
          - type: ejsTemplate
            data: # ejs □□□□□□□□ `<%= data.xx %>` □□
        frontmatterMode: merge # □□□□□□□□□□ frontmatter □□□□□□ ignore□□□□□ ign
```

frontmatterMode

- `ignore` □□□□□□□□□□ frontmatter□□□□□□□□□□□□□□ frontmatter

- `merge` □□□□□□□□□□ frontmatter□□□□□□□□□ key□□□□□□□□□□□□□□□□□□□□□□□

- `replace` □□□□□□□□□□ frontmatter □□□□□□□□□ frontmatter

- `remove` □□□□□□□□□□ frontmatter

□□□□□□□□□□□□□

□□

□□

□□□□□

□□

```
doom build  # □□□□□□
doom serve  # □□□□□□□□□□□
```

□□□□

□□ ci.yaml □□□□□□□□□□□□□□ Dockerfile □□□□□□□□□□□□

```
FROM build-harbor.alauda.cn/ops/alpine:latest
```

```
WORKDIR /docs

COPY . dist
```

□□□ ACP

□□□□□

□□□□□□□ `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

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Redirecting...</title>
    <meta http-equiv="refresh" content="0; url=/console-docs/4.1" />
  </head>
  <body>
    <p>Redirecting to <a href="/console-docs/4.1">/console-docs/4.1</a></p>
  </body>
</html>
```

□□□□□□□□

overrides.yaml

```yaml
# □□□□□□□□□□ console-platform □□□□□□□□□□□□□□□□□□□□
# https://gitlab-ce.alauda.cn/idp/Doom/-/blob/master/src/terms.ts#L11
terms:
  company:
    en: Alauda
    zh: □□□
  product:
    en: Alauda Container Platform
    zh: □□□□□□□
  productShort:
    en: ACP

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

```
en: Doom - Alauda
zh: Doom - □□□
```

versions.yaml

```
- '4.1'
- '4.0'
```

□□□□□□□□

□□□□□□□□ chart-frontend □□□□□□□□□□□□□□□□□□□□□□□□□ alauda-docs □□□ `check-alauda-docs` □□□□□□□□□□

□□□□□□□□

□□□□□□□□□□□□□□□□□□□□ `doom` □□□□□□□□□ idp □□□□ webapp □□□

> **INFO**
>
> □□□□□ PR □□□gitops □□□□□□