Introduction

Doom is a documentation development tool designed for internal use at Alauda, built on top of rspress. It provides users with a rich set of built-in plugins for an out-of-the-box experience.

TOC

Core Capabilities

- Automatically generates a configurable weight (order) left sidebar
- Full-text static document search
- Multilingual support

Based on Markdown and its extension MDX

MDX is a powerful content development approach. You can write Markdown files as you normally would while also using React components within the Markdown content:

For more details, you can refer to the “Using MDX” documentation↗.

Get Started

Let's get started quickly with Doom!

Getting Started

Getting Started

Creating a Project

CLI Tool

Usage

Configuration

Configure the `doom` documentation tool

Configuration File

Basic Configuration

API Documentation Configuration

Permission Documentation Configuration

Reference Documentation Configuration

Release Notes Configuration

Sidebar Configuration

Internal Document Routes Configuration

Only Include Document Routes Configuration

Syntax Highlighting Plugin Configuration

`sites.yaml` Configuration

Translation Configuration

Editing Documentation in Code Repository

Documentation Linting Configuration

Algolia Search Configuration

Sitemap Configuration

Convention

Based on the principle of "convention over configuration", we agree on the organization of documents to automatically generate the left sidebar and related content.

Directory Structure

Metadata

Sorting

Preview

Markdown

Callouts

Mermaid

MDX

Dynamic content display and content reuse can be achieved using MDX

rspress Components

doom Components

Custom Component Reuse

Internationalization

Using Internationalized Text in Reusable Components

`i18n.json`

`.ts/.tsx`

`.mdx`

API Documentation

Advanced APIs

CRD

Common References

Specifying OpenAPI Path

Permission Description Document

`props`

Example

Referencing Documents

Document Reference Configuration

Deployment

After completing the project development, we can deploy the project to the ACP platform.

Build and Preview

Multi-Version Builds

Merged Directory Structure

Dynamic Mounting Configuration File

Getting Started

TOC

Creating a Project

First, you can create a new directory with the following command:

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

Run `npm init -y` to initialize a project. You can install doom using npm, yarn, or pnpm:

npm yarn pnpm bun

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

Then create files with the following commands:

```
# Create docs directories, default supports bilingual Chinese and English
mkdir docs/en && echo '# Hello World' > docs/en/index.md
mkdir docs/zh && echo '# □□□□' > docs/zh/index.md
```

Add the following scripts to your `package.json`:

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

Then initialize a configuration file `doom.config.yml`:

```
title: My Docs
```

Also create a `tsconfig.json` file with the following content:

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

Finally, create a `global.d.ts` file with the following content:

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

This allows you to safely use the global components provided by doom in `.mdx` files with type safety.

CLI Tool

```
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
  -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
```

For more configuration, please refer to Configuration

Starting the Development Server

Run `yarn dev` to start the development server, the browser will automatically open the documentation homepage.

```
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
```

Production Build

Run `yarn build` to build the production code. After building, static files will be generated in the `dist` directory.

Local Preview

Run `yarn serve` to preview the built static files. Note that if you used `-b` or `-p` parameters during build, you need to use the same `-b` and `-p` parameters during preview.

Using Scaffolding Templates

Run `yarn new` to generate projects, modules, or documentation using scaffolding templates.

Translating Documentation

```
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]     Whether to copy relative assets to the target dire
  -h, --help               display help for command
```

- The `-g, --glob` parameter is required. You can specify the directories or paths of files to translate, supporting `glob` syntax. Note that the parameter value must be quoted to avoid unexpected behavior from command line parsing. Examples:

  1. `yarn translate -g abc xyz` will translate all documents under `<root>/<source>/abc` and `<root>/<source>/xyz` to `<root>/<target>/abc` and `<root>/<target>/xyz` respectively.
  2. `yarn translate -g '*'` will translate all documents under `<root>/<source>`.

- The `-C, --copy` parameter is optional. It controls whether to copy local asset files to the target directory when the target file does not exist. The default is `false`, which means changing the asset reference path to the source path. Examples:

  - When this parameter is enabled:

    1. When translating `/<source>/abc.jpg`, `<root>/public/<source>/abc.jpg` will be copied to `<root>/public/<target>/abc.jpg`, and the reference path in the document will be changed to `/<target>/abc.jpg`.
    2. In `<root>/<source>/abc.mdx`, the `./assets/xyz.jpg` reference will copy `<root>/<source>/assets/xyz.jpg` to `<root>/<target>/assets/xyz.jpg`, and the image reference path remains unchanged.

3. In `<root>/<source>/abc.mdx`, the `./assets/<source>/xyz.jpg` reference will copy `<root>/<source>/assets/<source>/xyz.jpg` to `<root>/<target>/assets/<target>/xyz.jpg`, and the reference path in the document will be changed to `./assets/<target>/xyz.jpg`.

- When this parameter is not enabled:

1. When translating `/<source>/abc.jpg`, if `<root>/public/<target>/abc.jpg` exists, the reference path in the document will be changed to `/<target>/abc.jpg`; otherwise, the image reference path remains unchanged.
2. In `<root>/<source>/abc.mdx`, if `<root>/<target>/assets/<target>/xyz.jpg` exists, the reference path `./assets/<source>/xyz.jpg` will be changed to `./assets/<target>/xyz.jpg`; otherwise, it will be changed to `../<source>/assets/<target>/xyz.jpg`.

WARNING

Specifically, if you use `-g '*'` for full translation, the file lists of the `source` and `target` directories will be compared, and unmatched `target` files except for `internalRoutes` will be automatically deleted.

TIP

The translation feature requires the local environment variable `AZURE_OPENAI_API_KEY` to be configured. Please contact your team leader to obtain it.

You can control translation behavior in the document metadata:

```yaml
i18n:
  title:
    en: DevOps Connectors
  additionalPrompts: 'The Connectors in this document are proper nouns and sh
  disableAutoTranslation: false
title: DevOps Connectors
```

For more configuration, please refer to Translation Configuration

Exporting PDF

> **WARNING**
>
> Please run the `yarn build` command before exporting.

```
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
```

Run `yarn export` to export the documentation as PDF files. Note that if you used `-b` or `-p` parameters during build, you need to use the same `-b` and `-p` parameters during export.

The export feature depends on `playwright` ↗. For CI pipelines, please use `build-harbor.alauda.cn/frontend/playwright-runner:doom` as the base image for dependency installation and documentation building. Locally, you can set the following environment variable to speed up downloads:

.env.yarn

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

Documentation Linting

```
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
```

For more configuration, please refer to Lint Configuration

Menu

Usage

Configuration

Configure the `doom` documentation tool

Convention

Based on the principle of "convention over configuration", we agree on the organization of documents to automatically generate the left sidebar and related content.

Directory Structure

Metadata

Sorting

Preview

Markdown

Callouts

Mermaid

MDX

Dynamic content display and content reuse can be achieved using MDX

rspress Components

doom Components

Custom Component Reuse

Internationalization

Using Internationalized Text in Reusable Components

`i18n.json`

`.ts/.tsx`

`.mdx`

API Documentation

Advanced APIs

CRD

Common References

Specifying OpenAPI Path

Permission Description Document

`props`

Example

Referencing Documents

Document Reference Configuration

Deployment

After completing the project development, we can deploy the project to the ACP platform.

Build and Preview

Multi-Version Builds

Merged Directory Structure

Dynamic Mounting Configuration File

Configuration

TOC

Configuration File

In most cases, we only need to use a static `yaml` configuration file, supporting `doom.config.yaml` or `doom.config.yml`. For complex scenarios, such as requiring

dynamic configuration or custom `rspress` plugins, `js/ts` configuration files can be used, supporting multiple file formats including `.js/.ts/.mjs/.mts/.cjs/.cts`.

For `js/ts` configuration files, we need to export the configuration. You can use the `defineConfig` function exported from `@alauda/doom/config` to enable type assistance:

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

export default defineConfig({})
```

Basic Configuration

- `lang`: Default document language. To accommodate most projects, we support both Chinese and English documents by default. The default language is `en`. If the current documentation project does not require multilingual support, this can be set to `null` or `undefined`.
- `title`: Document title, displayed on the browser tab.
- `logo`: Logo at the top left of the document, supports image URLs or file paths. Absolute paths refer to files under the `public` directory, relative paths refer to files relative to the current tool directory. The default is the Alauda logo built into the `doom` package.
- `logoText`: Document title, displayed next to the logo at the top left.
- `icon`: Document favicon, defaults to the same as `logo`.
- `base`: Base path of the document, used when deploying to a non-root path, e.g., `product-docs`. Defaults to `/`.
- `outDir`: Build output directory, defaults to `dist/{base}/{version}`. If specified, it changes to `dist/{outDir}/{version}`, where `version` is optional. See Multi-version Build for reference.

API Documentation Configuration

```yaml
api:
  # CRD definition file paths, relative to the directory where doom.config.*
  crds:
    - docs/shared/crds/*.yaml
  # OpenAPI definition file paths, relative to the directory where doom.confi
  openapis:
    - docs/shared/openapis/*.json
  # When rendering OpenAPI related resource definitions, they are inlined by
  # Reference https://doom.alauda.cn/apis/references/CodeQuality.html#v1alpha
  references:
    v1alpha1.CodeQualityBranch: /apis/references/CodeQualityBranch#v1alpha1.C
  # Optional, API documentation path prefix. If the current business uses gat
  pathPrefix: /apis
```

Refer to API Documentation for writing documentation.

Permission Documentation Configuration

```yaml
# The following resource file paths are relative to the directory where doom.
permission:
  functionresources:
    # `kubectl get functionresources`
    - docs/shared/functionresources/*.yaml
  roletemplates:
    # `kubectl get roletemplates -l auth.cpaas.io/roletemplate.official=true`
    - docs/shared/roletemplates/*.yaml
```

Refer to Permission Documentation for writing documentation.

Reference Documentation Configuration

```
reference:
  - repo: alauda-public/product-doc-guide # Optional, referenced documentatio
    branch: # [string] Optional, branch of the referenced documentation repos
    publicBase: # [string] Optional, when using a remote repository, the abso
    sources:
      - name: anchor # Name of the referenced document, used for referencing
        path: docs/index.mdx#▫▫ # Path of the referenced document, supports a
        ignoreHeading: # [boolean] Optional, whether to ignore the heading. I
        processors: # Optional, processors for referenced document content
          - type: ejsTemplate
            data: # ejs template parameters, accessed via `<%= data.xx %>`
        frontmatterMode: merge # Optional, mode for handling frontmatter of r
```

`frontmatterMode`

- `ignore`: Ignore the frontmatter of the referenced document, keep using the current document's frontmatter.
- `merge`: Merge the frontmatter of the referenced document. If keys conflict, the referenced document's values override the current document's.
- `replace`: Replace the current document's frontmatter with that of the referenced document.
- `remove`: Remove the current document's frontmatter.

Refer to Reference Documentation for writing documentation.

Release Notes Configuration

```
releaseNotes:
  queryTemplates:
    fixed: # JQL statements that may contain ejs templates
    unfixed:
```

release-notes.md

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

release-notes.mdx

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

Taking `template=fixed&project=DevOps` as an example, `fixed` is the template name defined in `queryTemplates`. The remaining `query` parameter `project=DevOps` is passed as `ejs` ↗ template parameters to the `fixed` template, which after processing is used as a jira `jql` ↗ to initiate a request to `https://jira.alauda.cn/rest/api/2/search?jql=<jql>`. This API requires authentication, and the environment variables `JIRA_USERNAME` and `JIRA_PASSWORD` must be provided to preview successfully.

Sidebar Configuration

```
sidebar:
  collapsed: false # Optional, whether to collapse the sidebar by default. De
```

Internal Document Routes Configuration

```
internalRoutes: # Optional, supports glob patterns, relative to the docs dire
  - '*/internal/**'
```

Only Include Document Routes Configuration

```
onlyIncludeRoutes: # Optional, supports glob patterns, relative to the docs d
  - '*/internal/**'
internalRoutes:
  - '*/internal/overview.mdx'
```

Syntax Highlighting Plugin Configuration

```
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

Unconfigured languages will trigger warnings in the command line and fallback to `plaintext` rendering.

`sites.yaml` Configuration

The `sites.yaml` configuration file is used to configure subsite information associated with the current documentation site. This information is used by External Site Components and when building single-version documentation.

```
- name: connectors # Unique name across the entire site
  base: /devops-connectors # Base path for site access
  version: v1.1 # Version used for ExternalSite/ExternalSiteLink redirection

  displayName: # Site display name. If not filled or language not matched, de
    en: DevOps Connectors
    zh: DevOps ▫▫▫

  # The following properties are used to pull images when building the entire
  # Usually required for subsite references, not required for parent site ref
  repo: https://github.com/AlaudaDevops/connectors-operator # Site repository
  image: devops/connectors-docs # Site build image, used to pull images when
```

Translation Configuration

```
translate:
  # System prompt, ejs template, parameters passed include `sourceLang`, `tar
  # Where `sourceLang` and `targetLang` are the strings `▫▫` and `▫▫`,
  #     `userPrompt` is the global user configuration below, may be empty
  #     `additionalPrompts` is the `additionalPrompts` configuration in docum
  #     `terms` and `titleTranslationPrompt` are prompts dynamically generate
  # The default system prompt is as follows and can be modified according to
  systemPrompt: |
You are a professional technical documentation engineer, skilled in writing h

## Baseline Requirements
- Sentences should be fluent and conform to the expression habits of the <%=
- Input format is MDX; output format must also retain the original MDX format
- **CRITICAL**: Do not translate or modify ANY link content in the document.
  - URLs in markdown links: [text](URL) - keep URL exactly as is
  - Reference-style links: [text][ref] and [ref]: URL - keep both ref and URL
  - Inline URLs: https://example.com - keep completely unchanged
  - Image links: ![alt](src) - keep src unchanged, but alt text can be transl
  - Anchor links: [text](#anchor) - keep #anchor unchanged
  - Any href attributes in HTML tags - keep unchanged
- Do not translate professional technical terms and proper nouns, including b
- The title field and description field in frontmatter should be translated,
- Content within MDX components needs to be translated, whereas MDX component
- Do not modify or translate any placeholders in the format of __ANCHOR_N__ (
- Keep original escape characters like backslash, angle brackets, etc. unchan
- Do not add any escape characters to special characters like [], (), {}, etc
  - If source has "Architecture [Optional]", keep it as "Architecture [Option
  - If source has "Function (param)", keep it as "Function (param)" (not "Fun
  - Only add escape characters if they were present in the original text
- Preserve and do not translate the following comments, nor modify their cont
  - {/* release-notes-for-bugs */}
  - <!-- release-notes-for-bugs -->
- Remove and do not retain the following comments:
  - {/* reference-start */}
  - {/* reference-end */}
  - <!-- reference-start -->
  - <!-- reference-end -->
- Ensure the original Markdown format remains intact during translation, such
- Do not translate the content of the code block.
<% if (titleTranslationPrompt) { %>
<%- titleTranslationPrompt %>
<% } %>
<% if (terms) { %>
```

```
<%- terms %>
<% } %>

<% if (userPrompt || additionalPrompts) { %>
## Additional Requirements
These are additional requirements for the translation. They should be met alo

The text for translation is provided below, within triple quotes:
"""
<% if (userPrompt) { %>
<%- userPrompt %>
<% } %>

<% if (additionalPrompts) { %>
<%- additionalPrompts %>
<% } %>
"""
<% } %>
```

Editing Documentation in Code Repository

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

Documentation Linting Configuration

```
lint:
  cspellOptions: # Optional, cspell configuration options, refer to https://g
```

Algolia Search Configuration

```yaml
algolia: # Optional, Algolia search configuration, effective only when the CL
  appId: # Algolia Application ID
  apiKey: # Algolia API Key
  indexName: # Algolia index name
```

Please use `public/robots.txt` for Algolia crawler verification.

INFO

Due to current architectural limitations of `rspress`, using Algolia search requires implementing via custom themes↗. To unify usage of related theme features, we provide the `@alauda/doom/theme` theme entry. Please add the following theme configuration file to enable:

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

Sitemap Configuration

```yaml
siteUrl: https://docs.alauda.cn # Optional, site URL used for generating site
```

Convention

TOC

Directory Structure

The left sidebar is automatically generated based on the file directory structure, where the `index` file in the first-level directory acts as the document's homepage and will display as the first item in the left navigation. Subfolders can use `index.md` or `index.mdx` and define the first-level title to set the grouping title for the left sidebar. Other sub-documents will be automatically merged into the current group, and nested subfolders will follow the same rules.

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

We also agree that:

1. The `public` directory is used to store static resources such as images, videos, etc.
2. The `public/_remotes` directory is used to store static resources associated with remote reference documents. Please do not directly rely on resources from this directory; you may

add `*/public/_remotes` to `.gitignore` to prevent these from being committed to the code repository.

3. The `shared` directory is for storing common components, reusable documents, etc., and will not automatically generate document data.

Metadata

At the beginning of the document, you can define the document's metadata such as title, description, author, category, etc., through the `frontmatter`.

```
---
title: Title
description: Description
author: Author
category: Category
---
```

In the body of the document, when using `.mdx` files, you can access these metadata through `frontmatter` as described in MDX.

Sorting

Other documents, except for `index.md` or `index.mdx`, will be sorted by default according to their file names. You can customize the `weight` value in the `frontmatter` to adjust the order of documents in the left sidebar (the smaller the `weight` value, the higher the priority in sorting).

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

WARNING

Note: Currently, changes to the left navigation configuration require a service restart to take effect, and it is usually not necessary to pay too much attention during development.

Preview

Sometimes, we do not need to display special content on the group homepage. In this case, you can use `index.mdx` file and the `Overview` component to display the list of documents in the current group. This will showcase the titles, descriptions, and secondary title information of the grouped list file.

```
# Usage

<Overview />
```

You can refer to Usage for the effect.

Markdown

In addition to the standard gfm syntax, Doom has some built-in extended Markdown features.

TOC

Callouts

Source code annotation component

NOTE

1. Please use inline code comments according to the actual language, such as `;`, `%`, `#`, `//`, `/** */`, `--`, and `<!-- -->`.
2. If you need to treat it as a code comment, use `[\!code callout]` for escaping.
3. Sometimes, `:::callouts` may display incorrectly due to nested indentation; you can use `<div class="doom-callouts">` or `<Callouts>` component instead.

````
```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) 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 \
              + 8 MiB × (number of vCPUs) \
              + 16 MiB × (number of graphics devices) \
              + (additional memory overhead)
```

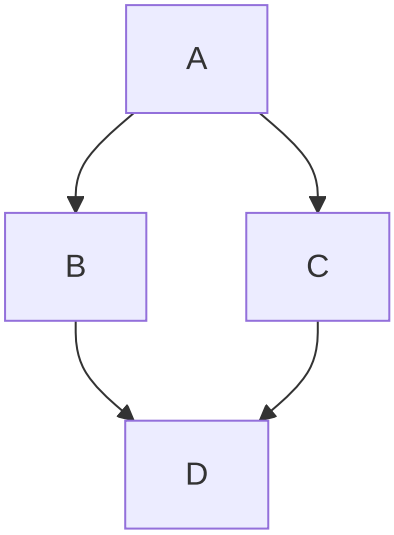

1. Required for the processes that run in the `virt-launcher` pod.
2. Number of virtual CPUs requested by the virtual machine.
3. Number of virtual graphics cards requested by the virtual machine.
4. Additional memory overhead:
   - If your environment includes a Single Root I/O Virtualization (SR-IOV) 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.

For more source code conversion features, please refer to Shiki Transformers↗.

Mermaid ↗

Chart drawing tool

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



Combined with Markdown Preview Mermaid ↗, you can preview in real-time within VSCode.

MDX

MDX is an extended syntax of Markdown that allows the use of JSX syntax within Markdown. For usage, you can refer to rspress MDX.

TOC

rspress Components

The `rspress` theme provides a majority of the built-in components as global components, which can be used directly in `.mdx` files without import, including:

- `Badge`
- `Card`
- `LinkCard`
- `PackageManagerTabs`
- `Steps`
- `Tab/Tabs`
- `Toc`

Other less frequently used components can be imported from `@rspress/core/theme`, for example:

preview.mdx

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

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

---

doom Components

`doom` provides some global components to assist in document writing, which can be used directly without import. Currently, these include:

`Overview`

Component for document overview, used to display the document directory.

`Directive`

Sometimes, due to nested indentation, the custom container↗ syntax may become invalid. The `Directive` component can be used as a substitute.

```
- The directory structure of multi-language documents (`doc/en`) needs to be

  <Directive type="danger" title="Attention">
    If automated translation tools are used for translation, there is no need
    worry about this issue, as the automated translation tools will
    automatically generate the directory structure of the target language
    documents based on `doc/zh`.
  </Directive>
```

- The directory structure of multi-language documents (`doc/en`) needs to be fully consistent with the documents under the `doc/zh` directory to ensure that the links of multi-language documents are identical except for the language identifier.

  Attention

  If automated translation tools are used for translation, there is no need to worry about this issue, as the automated translation tools will automatically generate the directory structure of the target language documents based on `doc/zh`.

`ExternalSite`

Component to reference external sites.

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

Note

Because DevOps Connectors releases on a different cadence from Alauda Container Platform, the DevOps Connectors documentation is now available as a separate documentation set at DevOps Connectors↗.

`ExternalSiteLink`

Component to reference external site links.

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

Content ↗

TIP

In mdx, `<ExternalSiteLink name="connectors" href="link" children="Content" />` has a different meaning from the content below

```
<ExternalSiteLink name="connectors" href="link">
  Content {/* this will be rendered inside a `p` element */}
</ExternalSiteLink>
```

If you do not want the text to be rendered inside a `p` element, you can use the `children` attribute as shown in the example above.

`AcpApisOverview` and `ExternalApisOverview`

Components to reference external site API overviews.

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

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

Note

For the introduction to the usage methods of ACP APIs, please refer to ACP APIs Guide ↗.

Note

For the introduction to the usage methods of DevOps Connectors APIs, please refer to DevOps Connectors APIs Guide ↗.

Term

Term component for plain text, dynamically mounted for injection.

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

Alauda alauda container platform ACP

`props`

- `name`: Built-in term name, refer to dynamic mounting configuration file.
- `textCase`: Text case transformation, optional values are `lower`, `upper`, `capitalize`.

`TermsTable`

Component for displaying a list of built-in terms.

```
<TermsTable />
```

Name	Chinese	Chinese Bad Cases	English	English Bad Cases	Description
company	□雀云	-	Alauda	-	公司品牌
product	□雀云容器平台	-	Alauda Container Platform	-	产品品牌
productShort	ACP	-	ACP	-	产品品牌简称

`props`

- `terms`: `NormalizedTermItem[]`, optional, a custom term list for reusing when rendering custom terms in internal documentation.

`JsonViewer`

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

yaml json

```
key: value
```

Custom Component Reuse

According to conventions, we can extract reusable content to the `shared` directory, then import it where needed, for example:

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

<CommonContent />
```

If you need to use more runtime ↗ related APIs, you can implement components using `.jsx/.tsx` and then import them into `.mdx` files.

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

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

WARNING

Note: Currently, components exported from `.mdx` do not support passing `props`. For scenarios where `props` need to be passed, please use `.jsx/.tsx` components for development, refer to this issue↗.

Internationalization

Most of the internal documentation for `alauda` is bilingual in Chinese and English. Therefore, we default to supporting the use of `en` / `zh` subfolders to store documentation in different languages. It is recommended to also store static resources in `en` / `zh` subfolders under the `public` directory, which facilitates the management of documentation content and static resources.

TOC

`i18n.json`

For reusable components that need to support both Chinese and English within the same component, you must first create an `i18n.json` file in the `docs` directory. Then, you can use `useI18n` in the component to retrieve the text in the current language, for example:

docs/i18n.json

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

Based on actual business needs, we generally categorize APIs into two types: Advanced APIs and CRDs (Custom Resource Definitions). Therefore, the directory structure is typically organized as follows:

```
├── apis
│   ├── advanced-apis # Advanced APIs
│   ├── crds # CRDs
│   └── references # Common References
```

TOC

Advanced APIs

advanced-apis/codeQualityTaskSummary.mdx

```
# CodeQualityTaskSummary

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

Refer to CodeQualityTaskSummary.

`props`

- `path`: The path under OpenAPI schema `paths`
- `pathPrefix`: Can be used to override the `api.pathPrefix` in global configuration
- `openapiPath`: Refer to Specifying OpenAPI Path

CRD

crds/ArtifactCleanupRun.mdx

```
# ArtifactCleanupRun

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

Refer to ArtifactCleanupRun.

`props`

- `name`: CRD `metadata.name`
- `crdPath`: Similar to Specifying OpenAPI Path, used to specify a particular CRD file

---

Common References

references/CodeQuality.mdx

```
# CodeQuality

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

Refer to CodeQuality.

`props`

- `schema`: The name under OpenAPI schema `definitions` (v2) or `components/schemas` (v3)
- `openapiPath`: Refer to Specifying OpenAPI Path

Specifying OpenAPI Path

For the `OpenAPIPath` and `OpenAPIRef` components, the default behavior is to search for matches across all OpenAPI definition files. If you need to specify a particular OpenAPI file, you can use the `openapiPath` property:

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

Permission Description Document

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

TOC

`props`

- `functions`: `string[]` - Required. An array of `FunctionResource` resource names to be displayed.

Example

Function	Action	Platform Administrator	Platform auditors	Project Manager	Namespace Administrato
testplans `devops-testplans`	View	✓	✓	✓	✓
	Create	✓	×	✓	✓
	Update	✓	×	✓	✓
	Delete	✓	×	✓	✓

Function	Action	Platform Administrator	Platform auditors	Project Manager	Namespace Administrato
testmodules `devops-testmodules`	View	✓	✓	✓	✓
	Create	✓	×	✓	✓
	Update	✓	×	✓	✓
	Delete	✓	×	✓	✓

Referencing Documents

In Markdown files:

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

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

In MDX files:

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

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

The `name` above refers to the name of the referenced document. For more information, please refer to Document Reference Configuration. If the referenced document content uses static resources from a remote repository, the related static resources will be automatically stored locally in the `<root>/public/_remotes/<name>` directory.

Here is an example using `<!-- reference-start#ref -->`:

TOC

Document Reference Configuration

```yaml
reference:
  - repo: alauda-public/product-doc-guide # Optional, repository address for
    branch: # [string] Optional, branch of the referenced document repository
    publicBase: # [string] Optional, the directory where static resources for
    sources:
      - name: anchor # Name of the referenced document, used to reference wit
        path: docs/index.mdx#introduction # Path to the referenced document,
        ignoreHeading: # [boolean] Optional, whether to ignore headings. If t
        processors: # Optional, processors for handling the content of the re
          - type: ejsTemplate
            data: # EJS template parameters, accessed via `<%= data.xx %>`.
        frontmatterMode: merge # Optional, mode for handling the frontmatter
```

`frontmatterMode`

- `ignore`: Ignores the frontmatter of the referenced document and retains the frontmatter of the current document.
- `merge`: Merges the frontmatter of the referenced document. If there are the same keys, the values from the referenced document will overwrite those in the current document.
- `replace`: Replaces the frontmatter of the current document with that of the referenced document.
- `remove`: Removes the frontmatter of the current document.

For writing documentation, refer to Document Reference.

Deployment

TOC

Build and Preview

Before deployment, we need to build the project for the production environment and preview it locally to ensure the project runs correctly:

```
doom build # Build static artifacts
doom serve # Preview the build artifacts in production mode
```

Multi-Version Builds

By default, `doom build` will output the build artifacts to the `dist` directory. If multiple versions of the documentation need to be built, you can specify the version number using the `-v` parameter, for example:

```
# Typically determined by the branch name, such as release-4.0 corresponding
doom build -v 4.0 # Build version 4.0, output artifacts to dist/4.0, document
doom build -v master # Build master version, output artifacts to dist/master,
doom build -v {other} # Build other versions, output artifacts to dist/{other

# unversioned and unversioned-x.y are special version numbers used for buildi
doom build -v unversioned # Build document without version prefix, output art
doom build -v unversioned-4.0 # Build document without version prefix but dis
```

Merged Directory Structure

```
├── 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

```
<!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>
```

Dynamic Mounting Configuration File

overrides.yaml

```
# Document information, each document can mount to override default configura
title:
  en: Doom - Alauda
  zh: Doom - □□□
logoText:
  en: Doom - Alauda
  zh: Doom - □□□
```

versions.yaml

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