Table of contents

Get Started

Introduction

Welcome to the home of your new documentation

Setting up

The first step to world-class documentation is setting up your editing environments.

Edit Your Docs

Get your docs set up locally for easy development

Preview Changes

Preview your changes before you push to make sure they're perfect

Make it yours

Update your docs to your brand and add valuable content for the best user conversion.

Customize Style

Reference APIs

Customize your docs to your company’s colors and brands

Automatically generate endpoints from an OpenAPI spec

Add Components

Build interactive features and designs to guide your users

Get Inspiration

Check out our showcase of our favorite documentation

Quickstart

Start building awesome documentation in under 5 minutes

Setup your development

Learn how to update your docs locally and deploy them to the public.

Edit and preview

Clone your docs locally

During the onboarding process, we created a repository on your Github with your docs content. You can find this repository on our **dashboard**. To clone the repository locally, follow these **instructions** in your terminal.

Preview changes

Previewing helps you make sure your changes look as intended. We built a command line interface to render these changes locally. 1. Install the **Mintlify CLI** to preview the documentation changes locally with this command: `npm i -g mintlify` 2. Run the following command at the root of your documentation (where `mint.json` is): `mintlify dev`

Deploy your changes

Install our Github app

Our Github app automatically deploys your changes to your docs site, so you don't need to manage deployments yourself. You can find the link to install on your **dashboard**. Once the bot has been successfully installed, there should be a check mark next to the commit hash of the repo.

Push your changes

Commit and push your changes to Git for your changes to update in your docs site. If you push and don't see that the Github app successfully deployed your changes, you can also manually update your docs through our **dashboard**.

Update your docs

Add content directly in your files with MDX syntax and React components. You can use any of our components, or even build your own.

Style Your Docs
Add flair to your docs with personalized branding.

Add API Endpoints
Implement your OpenAPI spec and enable API user interaction.

Integrate Analytics
Draw insights from user interactions with your documentation.

Host on a Custom Domain
Keep your docs on your own website's subdomain.

Development

Preview changes locally to update your docs

Prerequisite: Please install Node.js (version 19 or higher) before proceeding.

Follow these steps to install and run Mintlify on your operating system:

Step 1: Install Mintlify:

npm

```
npm i -g mintlify
```

yarn

```
yarn global add mintlify
```

Step 2: Navigate to the docs directory (where the `mint.json` file is located) and execute the following command:

```
mintlify dev
```

A local preview of your documentation will be available at `http://localhost:3000`.

Custom Ports

By default, Mintlify uses port 3000. You can customize the port Mintlify runs on by using the `--port` flag. To run Mintlify on port 3333, for instance, use this command:

```
mintlify dev --port 3333
```

If you attempt to run Mintlify on a port that's already in use, it will use the next available port:

```
Port 3000 is already in use. Trying 3001 instead.
```

Mintlify Versions

npm

```
npm i -g mintlify@latest
```

yarn

```
yarn global upgrade mintlify
```

npm yarn

```
npm i -g mintlify@latest
```

Validating Links

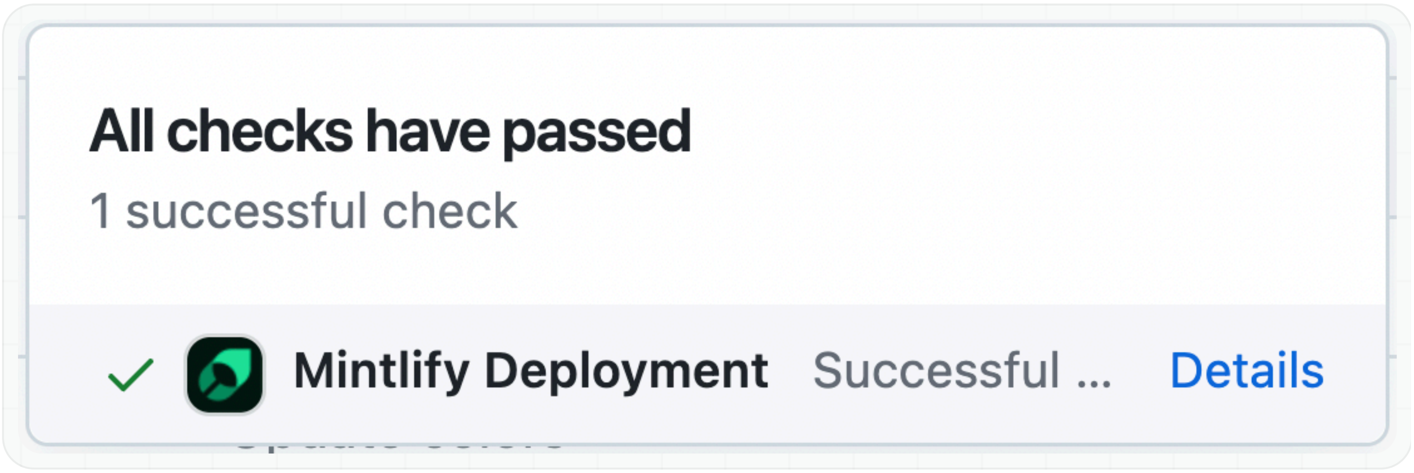
The CLI can assist with validating reference links made in your documentation. To identify any broken links, use the following command:

```
mintlify broken-links
```

Deployment

Unlimited editors available under the **Pro Plan** and above.

If the deployment is successful, you should see the following:


All checks have passed

1 successful check

✓ Mintlify Deployment Successful ... Details

Code Formatting

We suggest using extensions on your IDE to recognize and format MDX. If you're a VSCode user, consider the **MDX VSCode extension** for syntax highlighting, and **Prettier** for code formatting.

Troubleshooting

▾ Error: Could not load the "sharp" module using the darwin-arm64 runtime

This may be due to an outdated version of node. Try the following:

1. Remove the currently-installed version of mintlify: `npm remove -g mintlify`
2. Upgrade to Node v19 or higher.
3. Reinstall mintlify: `npm install -g mintlify`

▾ Issue: Encountering an unknown error

Solution: Go to the root of your device and delete the ~/.mintlify folder. Afterwards, run `mintlify dev` again.

Curious about what changed in the CLI version? **Check out the CLI changelog.**

Essentials

Markdown Syntax

Text, title, and styling in standard markdown

Titles

Best used for section headers.

```
## Titles
```

Subtitles

Best use to subsection headers.

```
### Subtitles
```

Each **title** and **subtitle** creates an anchor and also shows up on the table of contents on the right.

Text Formatting

We support most markdown formatting. Simply add `**`, `_`, or `~` around text to format it.

Style	How to write it	Result
Bold	`**bold**`	**bold**
Italic	`_italic_`	*italic*
Strikethrough	`~strikethrough~`	~~strikethrough~~

You can combine these. For example, write `**_bold and italic_**` to get ***bold and italic*** text.

You need to use HTML to write superscript and subscript text. That is, add `<sup>` or `<sub>` around your text.

Text Size	How to write it	Result
Superscript	`<sup>superscript</sup>`	$^{\text{superscript}}$
Subscript	`<sub>subscript</sub>`	$_{\text{subscript}}$

Linking to Pages

You can add a link by wrapping text in `[]()`. You would write `[link to google](https://google.com)` to link to google.

Links to pages in your docs need to be root-relative. Basically, you should include the entire folder path. For example, `[link to text](/writing-content/text)` links to the page "Text" in our components section.

Relative links like `[link to text](../text)` will open slower because we cannot optimize them as easily.

Blockquotes

Singleline

To create a blockquote, add a `>` in front of a paragraph.

> Dorothy followed her through many of the beautiful rooms in her castle.

```
> Dorothy followed her through many of the beautiful rooms in her castle.
```

Multiline

> Dorothy followed her through many of the beautiful rooms in her castle. The Witch bade her clean the pots and kettles and sweep the floor and keep the fire fed with wood.

```
> Dorothy followed her through many of the beautiful rooms in her castle.
>
> The Witch bade her clean the pots and kettles and sweep the floor and keep the f
```

LaTeX

Mintlify supports **LaTeX** through the Latex component.

8 x (vk x H1 - H2) = (0,1)

```
<Latex>8 x (vk x H1 - H2) = (0,1)</Latex>
```

Code Blocks

Display inline code and code blocks

Basic

Inline Code

To denote a `word` or `phrase` as code, enclose it in backticks (`).

```
To denote a `word` or `phrase` as code, enclose it in backticks (`).
```

Code Block

Use **fenced code blocks** by enclosing code in three backticks and follow the leading ticks with the programming language of your snippet to get syntax highlighting. Optionally, you can also write the name of your code after the programming language.

HelloWorld.java

```java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

~~~
```java HelloWorld.java
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```
~~~

Essentials

Images and Embeds

Add image, video, and other HTML elements

Image

Using Markdown

The **markdown syntax** lets you add images using the following code

```
![title](/path/image.jpg)
```

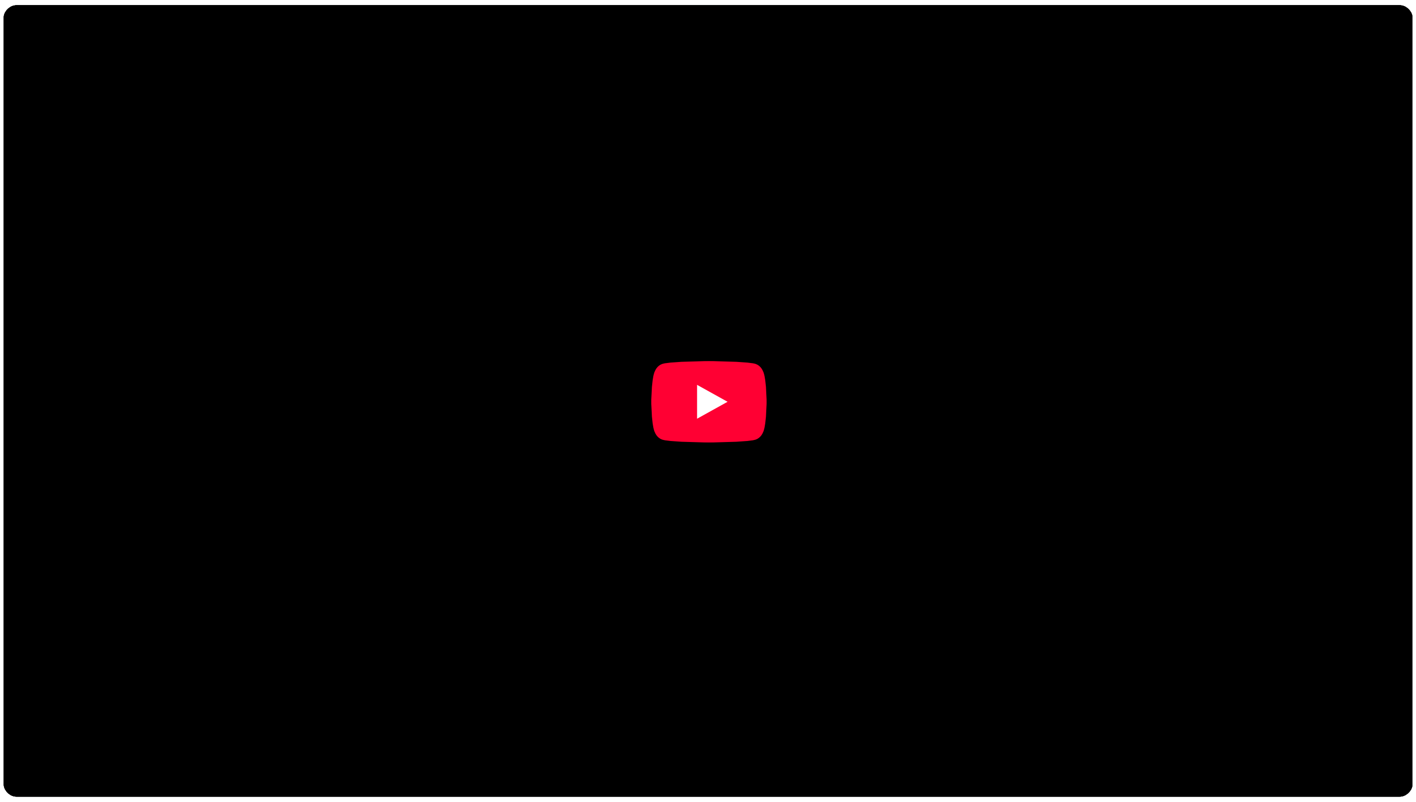
Note that the image file size must be less than 5MB. Otherwise, we recommend hosting on a service like **Cloudinary** or **S3**. You can then use that URL and embed.

Using Embeds

To get more customizability with images, you can also use **embeds** to add images

```
<img height="200" src="/path/image.jpg" />
```

Embeds and HTML elements



Mintlify supports **HTML tags in Markdown**. This is helpful if you prefer HTML tags to Markdown syntax, and lets you create documentation with infinite flexibility.

iFrames

Loads another HTML page within the document. Most commonly used for embedding videos.

```
<iframe src="https://www.youtube.com/embed/4KzFe50RQkQ"> </iframe>
```

Settings

Configure the global settings for your documentation

The docs.json file
Skip to the reference

Every documentation site requires a **docs.json** file.

This file contains the global configuration settings and controls everything from styling and navigation to integrations.

Reference

This section contains the full reference for the docs.json file.

Customization

`theme` `required`

One of the following: `mint` , `maple` , `palm` , `willow` , `linden` , `almond` .

The layout theme of the project. Check out the Themes page for more information.

`name` `string` `required`

The name of the project, organization, or product

`colors` `object` `required`

The colors to use in your documentation. At the very least, you must define the primary color. For example:

```json
{
  "colors": {
    "primary": "#ff0000"
  }
}
```

Hide Colors

`primary` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$` `required`

The primary color of the theme

Must be a hex code beginning with `#`

`light` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$`

The light color of the theme. Used for dark mode

Must be a hex code beginning with `#`

`dark` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$`

The dark color of the theme. Used for light mode

Must be a hex code beginning with `#`

`description` `string`

Optional description used for SEO and LLM indexing

`logo` `string or object`

The logo (for both light and dark mode)

Hide Logo

`light` `string` `required`

Path pointing to the light logo file to use in dark mode, including the file extension. Example: `/logo.png`

`dark` `string` `required`

Path pointing to the dark logo file to use in light mode, including the file extension. Example: `/logo-dark.png`

`href` `string (uri)`

The URL to redirect to when clicking the logo. If not provided, the logo will link to the homepage. Example: `https://example.com`

`favicon` `string or object`

The path to your favicon file in the docs folder, including the file extension. The file will automatically be resized to appropriate favicon sizes. Can be a single file or a pair for light and dark mode. Example: `/favicon.png`

Hide Favicon

`light` `string` `required`

Path pointing to the light favicon file to use in dark mode, including the file extension. Example: `/favicon.png`

`dark` `string` `required`

Path pointing to the dark favicon file to use in light mode, including the file extension. Example: `/favicon-dark.png`

`styling` `object`

Styling configurations

Hide Styling

`eyebrows` `"section" | "breadcrumbs"`

The eyebrows style of the content. Defaults to `section` .

`codeblocks` `"system" | "dark"`

The codeblock theme. Defaults to `system` .

`icons` `object`

Icon library settings

Hide Icons

`library` `"fontawesome" | "lucide"` `required`

The icon library to be used. Defaults to `fontawesome` .

`fonts` `object`

Hide Fonts

`family` `string` `required`

The font family, such as “Open Sans”, “Playfair Display”

`weight` `number`

The font weight, such as 400, 700. Precise font weights such as 550 are supported for variable fonts.

`source` `string (uri)`

The font source, such as https://mintlify-assets.b-cdn.net/fonts/Hubot-Sans.woff2

`format` `"woff" | "woff2"`

The font format, can be one of woff, woff2 `heading` `object`

Hide Heading

`family` `string` `required`

The font family, such as “Open Sans”, “Playfair Display”

`weight` `number`

The font weight, such as 400, 700. Precise font weights such as 550 are supported for variable fonts.

`source` `string (uri)`

The font source, such as https://mintlify-assets.b-cdn.net/fonts/Hubot-Sans.woff2

`format` `"woff" | "woff2"`

The font format, can be one of woff, woff2

`body` `object`

Hide Body

`family` `string` `required`

The font family, such as “Open Sans”, “Playfair Display”

`weight` `number`

The font weight, such as 400, 700. Precise font weights such as 550 are supported for variable fonts.

`source` `string (uri)`

The font source, such as https://mintlify-assets.b-cdn.net/fonts/Hubot-Sans.woff2 `format` `"woff" | "woff2"`

The font format, can be one of woff, woff2

`appearance` `object`

Light / dark mode toggle settings

Hide Appearance

`default` `"system" | "light" | "dark"`

The default light/dark mode. Defaults to `system`

`strict` `boolean`

Whether to hide the light / dark mode toggle. Defaults to `true` .

`background` `object`

Background color and decoration settings

Hide Background

`image` `string or object`

Hide Image

`light` `string` `required`

`dark` `string` `required`

`decoration` `"gradient" | "grid" | "windows"`

The background decoration of the theme

`color` `object`

The colors of the background

Hide Color

`light` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$`

The color in hex format to use in light mode

Must match pattern: ^#([a-fA-F0-9]6|[a-fA-F0-9]3)$

`dark` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$`

The color in hex format to use in dark mode

Must match pattern: ^#([a-fA-F0-9]6|[a-fA-F0-9]3)$

Structure

`navbar` `object`

Navbar content and settings

Hide Navbar

`links` `array of object`

The links in the navbar

Hide Links

`label` `string` `required`

`href` `string (uri)` `required`

A valid path or external link

`primary` `object`

Hide Primary

`type` `"button" | "github"` `required`

`label` `string` `required`

The label for the primary button. This only applies when `type` is set to `button`.

`href` `string (uri)` `required`

A valid path or external link. If `type` is set to `github`, this will be the URL to the repository.

`navigation` `object` `required`

The navigation structure of the content

Hide Navigation

`global` `object`

Add external links that will appear on all sections and pages irregardless of navigation nesting

Hide Global

`languages` `array of object`

Hide Languages

`language`

`"en" | "cn" | "zh" | "zh-Hans" | "zh-Hant" | "es" | "fr" | "ja" | "jp" | "pt" | "pt-BR" | "de" | "ko" | "it" | "ru" | "id" | "ar" | "tr"` `required`

The name of the language in the ISO 639-1 format

`default` `boolean`

Whether this language is the default language

`hidden` `boolean`

Whether the current option is default hidden

`href` `string (uri)` `required`

A valid path or external link

`versions` `array of object`

Hide Versions

`version` `string` `required`

The name of the version

Minimum length: 1

`default` `boolean`

Whether this version is the default version

`hidden` `boolean`

Whether the current option is default hidden

`href` `string (uri)` `required`

An external link

`tabs` `array of object`

Hide Tabs

tab `string` `required`

The name of the tab

Minimum length: 1

icon `string or object`

The icon to be displayed in the section

hidden `boolean`

Whether the current option is default hidden

href `string (uri)` `required`

An external link

anchors `array of object`

Hide Anchors

anchor `string` `required`

The name of the anchor

Minimum length: 1

icon `string or object`

The icon to be displayed in the section

color `object`

Hide Color

`light` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$`

The color in hex format to use in light mode

Must match pattern: ^#([a-fA-F0-9]6|[a-fA-F0-9]3)$

`dark` `string matching ^#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$`

The color in hex format to use in dark mode

Must match pattern: ^#([a-fA-F0-9]6|[a-fA-F0-9]3)$

`hidden` `boolean`

Whether the current option is default hidden

`href` `string (uri)` `required`

A valid path or external link

`dropdowns` `array of object`

Hide Dropdowns

`dropdown` `string` `required`

The name of the dropdown

Minimum length: 1

`icon` `string or object`

The icon to be displayed in the section

`hidden` `boolean`

Whether the current option is default hidden

`href` `string (uri)` `required`

An external link

`languages` `array of object`

Organizing by **languages**

`versions` `array of object`

Organizing by **versions**

`tabs` `array of object`

Organizing by **tabs**

`anchors` `array of object`

Organizing by **anchors**

`dropdowns` `array of object`

Organizing by **dropdowns**

`groups` `array of object`

Organizing by **groups**

`pages` `array of string or object`

An array of **page paths or groups**

`footer` `object`

Footer configurations

Hide Footer

`socials` `object`

An object in which each key is the name of a social media platform, and each value is the url to your profile. For example:

```
{
  "x": "https://x.com/mintlify"
}
```

Valid property names: `x` , `website` , `facebook` , `youtube` , `discord` , `slack` , `github` , `linkedin` , `instagram` , `hacker-news` , `medium` , `telegram` , `twitter` , `x-twitter` , `earth-americas` , `bluesky` , `threads` , `reddit` , `podcast`

`links` `array of object`

The links to be displayed in the footer

Hide Links

`header` `string`

The header title of the column

Minimum length: 1

`items` `array of object` `required`

The links to be displayed in the column

Hide Items

`label` `string` `required`

The label of the link

Minimum length: 1

`href` `string (uri)` `required`

The url of the link

banner `object`

Banner configurations

Hide Banner

content `string`

The content of the banner. This can be a string of text or a markdown string. For example:

```
{
  "content": "🚀 Banner is live! [Learn more](mintlify.com)"
}
```

dismissible `boolean`

Whether the banner is dismissible. Defaults to `false`.

redirects `array of object`

Hide Redirects

source `string` `required`

destination `string` `required`

permanent `boolean`

contextual `object`

Hide Contextual

`options` `array of "copy" | "view" | "chatgpt" | "claude"` `required`

The options to be displayed in the contextual menu. The first option is the default option.

- `copy` : Copy the current page as markdown to the clipboard
- `view` : View the current page as markdown in a new tab
- `chatgpt` : Feed the current page to ChatGPT
- `claude` : Feed the current page to Claude

Contextual Menu

⚠ The contextual menu is only available on preview & production deployments.

API Configurations

`api` `object`

API reference configuration and playground settings

Hide Api

`openapi` `string or array or object`

A string or an array of strings of absolute or relative urls pointing to the OpenAPI file(s)

Hide Openapi

`source` `string`

Minimum length: 1

`directory` `string`

no starting slash in the directory

`asyncapi` `string or array or object`

A string or an array of strings of absolute or relative urls pointing to the AsyncAPI file(s)

Hide Asyncapi

`source` `string`

Minimum length: 1

`directory` `string`

`playground` `object`

Configurations for the API playground

Hide Playground

`display` `"interactive" | "simple" | "none"`

The display mode of the API playground. Defaults to `interactive`.

`proxy` `boolean`

Whether to pass API requests through a proxy server. Defaults to `true`.

`examples` `object`

Configurations for the autogenerated API examples

Hide Examples

`languages` `array of string`

Example languages for the autogenerated API snippets

`defaults` `"required" | "all"`

Whether to show optional parameters in api examples, defaults to `all`

`mdx` `object`

Configurations for API pages generated from MDX files

Hide Mdx

`auth` `object`

Authentication configuration for the API

Hide Auth

`method` `"bearer" | "basic" | "key" | "cobo"`

Authentication method for the API

`name` `string`

Authentication name for the API

`server` `string or array`

SEO & Search

`seo` `object`

SEO indexing configurations

Hide Seo

`metatags` `object`

Meta tags added to every page. Must be a valid key-value pair. Possible options here

`indexing` `"navigable" | "all"`

Specify which pages to be indexed by search engines. Setting `navigable` indexes pages that are set in navigation, `all` indexes all pages. Defaults to `navigable`.

`search` `object`

Search display settings

Hide Search

`prompt` `string`

The prompt to be displayed in the search bar placeholder

Integrations

`integrations` `object`

Configurations for official integrations

Hide Integrations

`amplitude` `object`

Hide Amplitude

`apiKey` `string` `required`

`clearbit` `object`

Hide Clearbit

`publicApiKey` `string` `required`

`fathom` `object`

Hide Fathom

`siteId` `string` `required`

`frontchat` `object`

Hide Frontchat

`snippetId` `string` `required`

Minimum length: 6

`ga4` `object`

Hide Ga4

`measurementId` `string matching ^G` `required`

Must match pattern: ^G

`gtm` `object`

Hide Gtm

`tagId` `string matching ^G` `required`

Must match pattern: ^G

`heap` `object`

Hide Heap

`appId` `string` `required`

`hotjar` `object`

Hide Hotjar

`hjid` `string` `required`

`hjsv` `string` `required`

`intercom` `object`

Hide Intercom

`appId` `string` `required`

Minimum length: 6

`koala` `object`

Hide Koala

`publicApiKey` `string` `required`

Minimum length: 2

`logrocket` `object`

Hide Logrocket

`appId` `string` `required`

`mixpanel` `object`

Hide Mixpanel

`projectToken` `string` `required`

`osano` `object`

Hide Osano

`scriptSource` `string` `required`

`pirsch` `object`

Hide Pirsch

`id` `string` `required`

`posthog` `object`

Hide Posthog

`apiKey` `string matching ^phc\_` `required`

Must match pattern: ^phc_

`apiHost` `string (uri)`

`plausible` `object`

Hide Plausible

`domain` `string` `required`

`server` `string`

`segment` `object`

Hide Segment

`key` `string` `required`

`telemetry` `object`

Hide Telemetry

`enabled` `boolean`

`cookies` `object`

Hide Cookies

`key` `string`

`value` `string`

Errors

`errors` `object`

Hide Errors

`404` `object`

Hide 404

`redirect` `boolean`

Whether to redirect to the home page, if the page is not found

Validation

It is advised to include the following schema reference at the top of your docs.json file to ensure proper validation while editing:

```json
{
  "$schema": "https://mintlify.com/docs.json",
  ...
}
```

Navigation

The navigation field in mint.json defines the pages that go in the navigation menu

The navigation menu is the list of links on every website.

You will likely update `mint.json` every time you add a new page. Pages do not show up automatically.

Navigation syntax

Our navigation syntax is recursive which means you can make nested navigation groups. You don't need to include `.mdx` in page names.

Regular Navigation

```
"navigation": [
    {
        "group": "Getting Started",
        "pages": ["quickstart"]
    }
]
```

Nested Navigation

```
"navigation": [
    {
        "group": "Getting Started",
        "pages": [
            "quickstart",
            {
                "group": "Nested Reference Pages",
                "pages": ["nested-reference-page"]
            }
        ]
    }
]
```

Folders

Simply put your MDX files in folders and update the paths in `mint.json`.

For example, to have a page at `https://yoursite.com/your-folder/your-page` you would make a folder called `your-folder` containing an MDX file called `your-page.mdx`.

⚠ You cannot use `api` for the name of a folder unless you nest it inside another folder. Mintlify uses Next.js which reserves the top-level `api` folder for internal server calls. A folder name such as `api-reference` would be accepted.

Navigation With Folder

```
"navigation": [
    {
        "group": "Group Name",
        "pages": ["your-folder/your-page"]
    }
]
```

Hidden Pages

MDX files not included in `mint.json` will not show up in the sidebar but are accessible through the search bar and by linking directly to them.

Reusable Snippets

Reusable, custom snippets to keep content in sync

One of the core principles of software development is DRY (Don't Repeat Yourself). This is a principle that apply to documentation as well. If you find yourself repeating the same content in multiple places, you should consider creating a custom snippet to keep your content in sync.

Creating a custom snippet

Pre-condition: You must create your snippet file in the `snippets` directory.

Any page in the `snippets` directory will be treated as a snippet and will not be rendered into a standalone page. If you want to create a standalone page from the snippet, import the snippet into another file and call it as a component.

Default export

1. Add content to your snippet file that you want to re-use across multiple locations. Optionally, you can add variables that can be filled in via props when you import the snippet.

snippets/my-snippet.mdx

```
Hello world! This is my content I want to reuse across pages. My keyword of the
day is {word}.
```

The content that you want to reuse must be inside the `snippets` directory in order for the import to work.

2. Import the snippet into your destination file.

destination-file.mdx

```
---
title: My title
description: My Description
---

import MySnippet from '/snippets/path/to/my-snippet.mdx';

## Header

Lorem impsum dolor sit amet.

<MySnippet word="bananas" />
```

Reusable variables

1. Export a variable from your snippet file:

snippets/path/to/custom-variables.mdx

```
export const myName = 'my name';

export const myObject = { fruit: 'strawberries' };
```

2. Import the snippet from your destination file and use the variable:

destination-file.mdx

```
---
title: My title
description: My Description
---

import { myName, myObject } from '/snippets/path/to/custom-variables.mdx';

Hello, my name is {myName} and I like {myObject.fruit}.
```

Reusable components

1. Inside your snippet file, create a component that takes in props by exporting your component in the form of an arrow function.

snippets/custom-component.mdx

```
export const MyComponent = ({ title }) => (
  <div>
    <h1>{title}</h1>
    <p>... snippet content ...</p>
  </div>
);
```

⚠ MDX does not compile inside the body of an arrow function. Stick to HTML syntax when you can or use a default export if you need to use MDX.

2. Import the snippet into your destination file and pass in the props

destination-file.mdx

```
---
title: My title
description: My Description
---

import { MyComponent } from '/snippets/custom-component.mdx';

Lorem ipsum dolor sit amet.

<MyComponent title={'Custom title'} />
```

Introduction

Example section for showcasing API endpoints

If you’re not looking to build API reference documentation, you can delete this section by removing the api-reference folder.

Welcome

There are two ways to build API documentation: **OpenAPI** and **MDX components**. For the starter kit, we are using the following OpenAPI specification.

Plant Store Endpoints

View the OpenAPI specification file

Authentication

All API endpoints are authenticated using Bearer tokens and picked up from the specification file.

```
"security": [
  {
    "bearerAuth": []
  }
]
```

Endpoint Examples

Get Plants

Returns all plants from the system that the user has access to

GET /plants

Try it ▶

cURL

cURL

```
curl --request GET \
  --url http://sandbox.mintlify.com/plants \
  --header 'Authorization: Bearer <token>'
```

200 400

```
[
  {
    "name": "<string>",
    "tag": "<string>"
  }
]
```

Authorizations

`Authorization` `string` `header` `required`

Bearer authentication header of the form `Bearer <token>`, where `<token>` is your auth token.

Query Parameters

`limit` `integer`

The maximum number of results to return

Response

200 ⌄ application/json

Plant response

`name` `string` `required`

The name of the plant

`tag` `string`

Tag to specify the type

Endpoint Examples

Create Plant

Creates a new plant in the store

POST /plants

Try it

cURL

```
curl --request POST \
  --url http://sandbox.mintlify.com/plants \
  --header 'Authorization: Bearer <token>' \
  --header 'Content-Type: application/json' \
  --data '{
  "name": "<string>",
  "tag": "<string>",
  "id": 123
}'
```

200 400

```
{
  "name": "<string>",
  "tag": "<string>"
}
```

Authorizations

`Authorization` string header required

Bearer authentication header of the form `Bearer <token>`, where `<token>` is your auth token.

Body

application/json

Plant to add to the store

`name` `string` `required`

The name of the plant

`id` `integer` `required`

Identification number of the plant

`tag` `string`

Tag to specify the type

Response

200 ⌄ application/json

plant response

`name` `string` `required`

The name of the plant

`tag` `string`

Tag to specify the type

Endpoint Examples

Delete Plant

Deletes a single plant based on the ID supplied

DELETE /plants/{id}   Try it ▶

cURL   cURL

```
curl --request DELETE \
  --url http://sandbox.mintlify.com/plants/{id} \
  --header 'Authorization: Bearer <token>'
```

204   400

```
This response has no body data.
```

Authorizations

`Authorization` `string` `header` `required`

Bearer authentication header of the form `Bearer <token>`, where `<token>` is your auth token.

Path Parameters

`id` `integer` `required`

ID of plant to delete