

[configuration](#)[yaml](#)

Configuration

Atlas reads project settings from `atlas.yml` in the repository root. Commit the file so local and automated builds use the same rules.

Minimal configuration

```
project:
  name: Atlas Sample
  source_dir: docs
  output_dir: site

validation:
  fail_on: error
  check_links: true
  require_front_matter: true

review:
  human_approval_required: true
```

Validate the configuration after each change:

```
atlas config check
```

Configuration keys

Key	Type	Default	Description
<code>project.source_dir</code>	String	<code>docs</code>	Directory containing Markdown source files.
<code>project.output_dir</code>	String	<code>site</code>	Directory containing generated output.
<code>validation.fail_on</code>	String	<code>error</code>	Lowest severity that fails the build.
<code>validation.check_links</code>	Boolean	<code>true</code>	Checks local and external links.
<code>validation.require_front_matter</code>	Boolean	<code>false</code>	Requires YAML metadata in every page.
<code>review.human_approval_required</code>	Boolean	<code>true</code>	Blocks publishing until a reviewer approves.

Allowed values for `validation.fail_on` are `info`, `warning`, `error`, and `never`.

Environment-specific values

Use environment variables for secrets and deployment-specific values:

```
publishing:
  target: github-pages
  repository: ${ATLAS_REPOSITORY}
  token: ${ATLAS_TOKEN}
```

Set the variables in the CI platform's secret store. Do not add tokens to `atlas.yml`, workflow logs, examples, or screenshots.

Precedence

Atlas resolves settings in this order, from highest to lowest priority:

1. Command-line options
2. Environment variables
3. `atlas.yml`
4. Built-in defaults

For example, `atlas validate --fail-on warning` overrides `validation.fail_on` for one run.

Danger

Never commit an actual API key. If a credential is exposed, revoke it immediately and remove it from the Git history according to your incident-response procedure.

Related information: [Security and governance](#).