How to Host a Zola-Generated Site on GitLab Pages

2025-08-26

How to host a Zola-generated site on GitLab Pages

Introduction

After some time without a personal blog, I decided to write down what I have learned and used. This site is mainly a personal collection of tasks, tools, and setups that I use regularly.

Over the years, I tried plain text, plain HTML, and several CMSes and frameworks such as WordPress, Django, Gatsby, and Hugo. For some company websites, I still use a CMS because many business users are not comfortable with Git and text editors, and a CMS offers a more user-friendly interface.

For my personal site, I chose Zola because I wanted a Rust-based **static site generator**. When I checked Jamstack Site Generators, I found a few Rust options :crab: and picked Zola.

Why GitLab Pages?

These days, hosting a static site for free is straightforward. Many **version control systems** offer a Pages feature, including GitLab Pages. Why GitLab and not GitHub, Codeberg, or something else? Simply put: I already had a GitLab account and I like the platform.

What do I publish?

After choosing the tooling, the next question was content. There are already many blogs out there :hand_over_mouth: on almost every topic, and more people now use AI to get answers quickly.

My honest answer is: “for my own laziness” :nerd_face:.

I had knowledge spread across multiple places, so I kept searching for where I had documented a particular command or setup. This blog is my way to keep that knowledge in one place that I can access from anywhere.

Summary

So far, the setup looks like this:

- Tool: zola
- Hosting: GitLab Pages
- Content: Notes for myself, and hopefully useful for others too

Let's start writing

zola

The Zola docs begin with an overview and the core commands:

```
zola init none-blog
cd none-blog
git remote add origin https://gitlab.com/aleks001/none-blog.git
# original theme before March 2026
# git submodule add https://codeberg.org/salif/linkita.git themes/linkita
git submodule add https://github.com/thomasweitzel/zolarwind.git themes/
zolarwind
```

The last command adds the **Zolarwind** theme.

Update (March 2026): I moved from Linkita to Zolarwind because I preferred the design.
The commented command above reflects my original setup when I first published this post.

After that, I could start writing. You can find my blog posts here, and the list will keep growing.

The site config is here.

Create Content

Based on what I learned from analytics tools, I always include front matter like this:

```
---
title: How to host a Zola-generated site on GitLab Pages
description: How to host a Zola-generated site on GitLab Pages
updated: 2025-08-26
date: 2025-08-26
taxonomies:
  categories:
  - zola
  - gitlab
  tags:
  - zola
  - gitlab
  - rust
---
```

Then I just create Markdown files and write.

GitLab Pages

To use GitLab Pages, you need a GitLab account and a repository. This is mine for this blog: https://gitlab.com/aleks001/none-blog/.

Zola provides a GitLab Pages deployment guide. I adapted my `.gitlab-ci.yml` based on that guide. I also added a compression step so content is served compressed to browsers, which saves bandwidth.

This is my `.gitlab-ci.yml`:

```
stages:
  - build
  - deploy

default:
  image: debian:stable-slim

variables:
  # Fetch theme submodules automatically.
  GIT_SUBMODULE_STRATEGY: "recursive"

  # Keep CI on the same major/minor Zola as local development.
  ZOLA_VERSION: "0.22.1"

  SITE_BASE_URL: "https://blog.none.at"
  COMPRESS_REGEX: ".*\\.(htm|html|xml|txt|text|js|css|svg)$"

pages-build:
  stage: build
  script:
    - |
      apt-get update
      DEBIAN_FRONTEND=noninteractive apt-get install --assume-yes --no-
install-recommends wget ca-certificates
      ZOLA_RELEASE_DIR="https://github.com/getzola/zola/releases/
download/v${ZOLA_VERSION}"
      ZOLA_TARBALL="zola-v${ZOLA_VERSION}-x86_64-unknown-linux-
gnu.tar.gz"
      echo "*******************************************************"
      echo "*"
      echo "* Using zola version: ${ZOLA_VERSION}"
      echo "* Retrieving zola tarball: ${ZOLA_TARBALL}"
      echo "* ..from: ${ZOLA_RELEASE_DIR}"
      echo "*"
      echo "*******************************************************"
      wget --no-verbose "${ZOLA_RELEASE_DIR}/${ZOLA_TARBALL}"
      tar -xzf "${ZOLA_TARBALL}"
      ./zola build --base-url "${SITE_BASE_URL}" --minify
  artifacts:
    paths:
      - public
    expire_in: 1 week

pages:
  stage: deploy
  pages: true
```

```
  dependencies:
    - pages-build
  script:
    - |
      echo "Deploy pages built in previous stage"
      apt-get update
      DEBIAN_FRONTEND=noninteractive apt-get install --assume-yes --no-
install-recommends brotli zstd
      find public -type f -regex "${COMPRESS_REGEX}" -exec gzip -f -k {}
\;
      find public -type f -regex "${COMPRESS_REGEX}" -exec brotli -f -k
{} \;
      find public -type f -regex "${COMPRESS_REGEX}" -exec zstd -f -k {}
\;
  artifacts:
    paths:
      - public
    expire_in: 1 week
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Custom domains

Because GitLab Pages supports custom domains, I configured the `blog` subdomain for `none.at`. That gives me a cleaner URL, as described in the GitLab Pages DNS records documentation.

Make it public

After these steps, I run the usual Git commands to publish changes:

- git add .
- git commit -s -m ‘Init’
- git push

I repeat this workflow for future posts with a proper commit message :smile:.