Building a Website with Hugo & GitHub Pages

A Complete Setup Guide

From selecting a template to deploying with a custom domain

Updated with real-world troubleshooting and project templates

Introduction

This guide will walk you through creating a website using Hugo (a static site generator) and deploying it to GitHub Pages for free hosting. This is an excellent alternative to paid services like Squarespace and Webflow.

What You'll Need:

- A computer with terminal/command line access
- A GitHub account (free)
- A domain name (optional, but recommended)
- A text editor (VS Code, Sublime Text, etc.)
- Basic familiarity with command line

Benefits:

- Free hosting via GitHub Pages
- No monthly fees (unlike Squarespace/Webflow)
- Fast, secure static websites
- Full control over your code
- Version control with Git

Part 1: Installing Hugo

Hugo is a static site generator written in Go. It's extremely fast and perfect for building blogs, portfolios, and documentation sites.

Installation by Operating System:

macOS:

```
brew install hugo
```

Windows:

1. Download Hugo from https://github.com/gohugoio/hugo/releases
2. Extract the executable
3. Add Hugo to your PATH

Or use a package manager like Chocolatey or Scoop

Linux:

```
sudo apt-get install hugo
```

or

```
sudo snap install hugo
```

Verify Installation:

```
hugo version
```

You should see version information displayed. If you get an error, Hugo may not be properly installed or added to your PATH.

Part 2: Choosing a Hugo Theme

Hugo themes are templates that provide the design and structure for your website. You can browse hundreds of free themes and use them as starting points.

Where to Find Themes:

- https://themes.gohugo.io/ - Official Hugo themes directory
- GitHub - Search for 'hugo theme'
- Hugo Theme Console (minimal design): https://github.com/mrmierzejewski/hugo-theme-console

Popular Theme Categories:

- Blog themes - For writing and articles
- Portfolio themes - For showcasing work
- Documentation themes - For technical documentation
- Business themes - For company websites
- Minimal themes - Clean, simple designs

Part 3: Creating Your Hugo Site

Follow these steps in order to avoid common errors.

Step 1: Create a New Site

```
hugo new site my-website
cd my-website
```

This creates a new directory with the basic Hugo structure.

Step 2: Initialize Git (IMPORTANT!)

```
git init
```

This initializes version control in your project. You MUST do this before adding a theme as a submodule.

Step 3: Add a Theme

Now that Git is initialized, you can add a theme as a submodule:

```
git submodule add https://github.com/username/theme-name.git
themes/theme-name
```

Example with Console theme:

```
git submodule add https://github.com/mrmierzejewski/hugo-theme-console.git
themes/hugo-theme-console
```

Common Error:

If you see 'fatal: not a git repository', you forgot to run 'git init' first. Go back to Step 2!

Step 4: Configure Your Site

Edit the hugo.toml (or config.toml) file in your project root. Add the theme configuration:

```
echo 'theme = "hugo-theme-console"' >> hugo.toml
```

Or manually edit hugo.toml to include:

```
baseURL = 'https://yourusername.github.io/'
languageCode = 'en-us'
title = 'My New Hugo Site'
theme = 'hugo-theme-console'
```

Most themes include an exampleSite folder with a sample hugo.toml file you can copy and customize.

Part 4: Adding Content

Creating Blog Posts:

```
hugo new posts/my-first-post.md
```

This creates a new Markdown file in content/posts/ with front matter (metadata).

Creating Project Pages:

```
hugo new projects/project-name.md
```

This creates a project page. Use this for portfolio projects, case studies, or work samples.

Project Page Template:

```
--- title: "Project Name" date: 2025-10-31 draft: false --- **Role:**
Creative Technologist **Client:** [Client Name] **Associated:**
[Studio/Company Name] **Team:** Person 1, Person 2, Person 3 **Location:**
City, Country **Date:** Month Year ## Overview Brief project introduction and
your role. ## Project Description Detailed description of what was created,
the goals, and the approach. ## Technology Stack - Technology 1 - Technology
2 - Technology 3 ## Key Features - Feature 1 - Feature 2 - Feature 3 ##
Outcomes Results, impact, or reception of the project. ## Acknowledgments
Thanks to collaborators and team members.
```

Front Matter Explained:

The section between --- markers is called 'front matter' and contains metadata:

- title: The page title displayed on your site
- date: Publication date (format: YYYY-MM-DD)
- draft: Set to 'true' to hide from production, 'false' to publish
- tags: Optional list of tags for categorization
- categories: Optional categories for organization

Adding Images:

1. Place images in the static/images/ folder
2. Reference them in Markdown: ![Alt text](/images/photo.jpg)
3. Or organize by project: /images/project-name/photo.jpg

Key Points:

- draft: true means the page won't appear in production builds
- Content is written in Markdown (.md files)
- Images go in the static/ folder
- Page structure follows the content/ directory structure
- Use hugo serve -D to preview drafts locally

Part 5: Testing Your Site Locally

Start the Hugo Development Server:

```
hugo serve -D
```

Open your browser and go to: http://localhost:1313

The server automatically reloads when you make changes to your files. This is perfect for developing and previewing your site before deployment.

What You Should See:

When hugo serve runs successfully, you'll see output like:

```
Watching for changes in /path/to/your/site/...
Web Server is available at http://localhost:1313/
Press Ctrl+C to stop
```

Common Development Commands:

- hugo serve - Start development server
- hugo serve -D - Include draft posts
- hugo - Build site for production (creates public/ folder)
- hugo new content/about.md - Create new content file
- Ctrl+C - Stop the development server

Part 6: Setting Up GitHub Pages

Step 1: Create a GitHub Repository

1. Log in to GitHub
2. Click the '+' icon and select 'New repository'
3. Name it: yourusername.github.io (replace 'yourusername' with your actual GitHub username)
4. Make it public
5. Do NOT initialize with README, .gitignore, or license

> Important: The repository MUST be named exactly yourusername.github.io for GitHub Pages to work automatically.

Step 2: Connect Your Local Repository

From your project directory (where you ran hugo new site):

```
git add .
git commit -m 'Initial commit'
git branch -M main
git remote add origin
https://github.com/yourusername/yourusername.github.io.git
git push -u origin main
```

> Replace 'yourusername' with your actual GitHub username in the URL above.

Part 7: Automating Deployment with GitHub Actions

GitHub Actions will automatically build and deploy your Hugo site whenever you push changes.

Step 1: Create Workflow Directory

Create the directory structure for GitHub Actions:

```
mkdir -p .github/workflows
```

Step 2: Create hugo.yml Workflow File

Create a file at .github/workflows/hugo.yml with the following content (see next page):

GitHub Actions Workflow Configuration:

```
name: Deploy Hugo site to Pages on: push: branches: ["main"]
workflow_dispatch: permissions: contents: read pages: write id-token: write
concurrency: group: "pages" cancel-in-progress: false defaults: run: shell:
bash jobs: build: runs-on: ubuntu-latest env: HUGO_VERSION: 0.128.0 steps: -
name: Install Hugo run: | wget -O hugo.deb https://github.com/gohugoio/hugo/r
eleases/download/v${HUGO_VERSION}/hugo_extended_${HUGO_VERSION}_linux-amd64.d
eb sudo dpkg -i hugo.deb - name: Checkout uses: actions/checkout@v4 with:
submodules: recursive - name: Setup Pages id: pages uses:
actions/configure-pages@v5 - name: Build with Hugo run: hugo --minify
--baseURL "${{ steps.pages.outputs.base_url }}/" - name: Upload artifact
uses: actions/upload-pages-artifact@v3 with: path: ./public deploy:
environment: name: github-pages url: ${{ steps.deployment.outputs.page_url }}
runs-on: ubuntu-latest needs: build steps: - name: Deploy to GitHub Pages id:
deployment uses: actions/deploy-pages@v4
```

Step 3: Enable GitHub Pages

1. Go to your repository on GitHub
2. Click Settings → Pages (in left sidebar)
3. Under 'Build and deployment', Source: select 'GitHub Actions'
4. Save the changes

Step 4: Push Your Workflow

```
git add .
git commit -m 'Add GitHub Actions workflow'
git push
```

Your site should now be live at https://yourusername.github.io within a few minutes! Check the 'Actions' tab in your GitHub repository to monitor the deployment.

Part 8: Adding a Custom Domain

If you own a domain name, you can use it instead of the yourusername.github.io address.

Step 1: Configure DNS Records

Log in to your domain registrar (Squarespace, GoDaddy, Namecheap, Google Domains, etc.) and add DNS records:

Option A: Using www subdomain (Recommended)

```
Add a CNAME record:
• Host/Name: www
• Points to/Value: yourusername.github.io
• TTL: 3600 (or default)
```

Option B: Using apex domain (no www)

```
Add four A records:
• Host/Name: @ (or leave blank)
• Add these four IP addresses (create 4 separate A records):
- 185.199.108.153
- 185.199.109.153
- 185.199.110.153
- 185.199.111.153
```

> Best Practice: Set up both! GitHub will automatically redirect between them.

Step 2: Configure Custom Domain in GitHub

1. Go to your repository → Settings → Pages
2. Under 'Custom domain', enter your domain (e.g., www.yourdomain.com)
3. Click Save
4. Wait a few minutes for DNS to propagate
5. Check the box for 'Enforce HTTPS' (this may take a few minutes to become available)

Step 3: Update hugo.toml

Change the baseURL in your hugo.toml file:

```
baseURL = 'https://www.yourdomain.com/'
```

Make sure to use https:// and include the trailing slash!

Step 4: Push Changes

```
git add .
git commit -m 'Update baseURL for custom domain'
git push
```

DNS changes can take 24-48 hours to propagate worldwide, but usually work within minutes to a few hours.

DNS Provider-Specific Notes:

- Squarespace: Go to Settings → Domains → [your domain] → DNS Settings
- GoDaddy: Go to My Products → Domains → [your domain] → DNS
- Namecheap: Go to Domain List → Manage → Advanced DNS
- Google Domains: Go to My domains → [your domain] → DNS

Part 9: Making Updates and Publishing New Content

Typical Workflow:

1. Create or edit content files locally
2. Test with hugo serve -D
3. When satisfied, set draft: false in the front matter
4. Commit changes: git add . && git commit -m 'Add new post'
5. Push to GitHub: git push
6. GitHub Actions automatically builds and deploys your site (takes 1-3 minutes)

Creating Different Content Types:

- Blog post: hugo new posts/my-post.md
- Project page: hugo new projects/project-name.md
- About page: hugo new about.md
- Custom page: hugo new [section]/[filename].md

Useful Git Commands Reference:

- git status - See what files changed
- git add . - Stage all changes
- git commit -m 'message' - Commit with a message
- git push - Push changes to GitHub
- git log - View commit history
- git pull - Get latest changes from GitHub

Part 10: Common Issues and Solutions

'fatal: not a git repository' when adding theme

• Solution: Run 'git init' first, then add the theme submodule
• This is the most common error for beginners!

Site doesn't build on GitHub

• Check the Actions tab for error messages
• Ensure hugo.yml workflow file is properly formatted
• Verify theme is properly added as a submodule
• Check that baseURL is correct in hugo.toml
• Make sure GitHub Pages is set to 'GitHub Actions' source

Theme doesn't appear

• Confirm theme = 'theme-name' matches the folder name in themes/
• Check that submodule was added correctly: git submodule status
• Try: git submodule update --init --recursive
• Verify the theme folder isn't empty

Custom domain not working

• Verify DNS records are correct (use a DNS checker tool online)
• Wait for DNS propagation (can take up to 48 hours)
• Check that CNAME file exists in your repository
• Ensure HTTPS is enforced in GitHub Pages settings
• Make sure baseURL in hugo.toml matches your domain

Images not showing

• Place images in static/ folder (not content/)
• Reference as /images/photo.jpg (starting with /)
• Check baseURL is correct and ends with /

• Verify image files were committed and pushed to GitHub

Posts not appearing

• Ensure draft: false in front matter
• Check date is not in the future
• Verify content is in the correct folder structure
• Make sure the theme supports the content type you're using

Hugo serve shows empty site

• Use hugo serve -D to include draft content
• Check if theme is properly configured in hugo.toml
• Verify content files are in content/ directory
• Look at theme's exampleSite for proper structure

Part 11: Additional Resources

Official Documentation:

- Hugo Documentation: https://gohugo.io/documentation/
- Hugo Quick Start: https://gohugo.io/getting-started/quick-start/
- Hugo Themes: https://themes.gohugo.io/
- GitHub Pages Docs: https://docs.github.com/en/pages
- GitHub Actions: https://docs.github.com/en/actions
- Markdown Guide: https://www.markdownguide.org/

Community and Support:

- Hugo Forum: https://discourse.gohugo.io/
- Hugo GitHub: https://github.com/gohugoio/hugo
- GitHub Community: https://github.community/
- Stack Overflow: Tag your questions with 'hugo'

Useful Tools:

- DNS Checker: https://dnschecker.org/ - Check DNS propagation
- Git Basics: https://git-scm.com/book/en/v2
- VS Code: https://code.visualstudio.com/ - Popular code editor
- GitHub Desktop: https://desktop.github.com/ - GUI for Git

Quick Reference Card

Complete Setup Sequence:

```
# 1. Create new site hugo new site my-website cd my-website # 2. Initialize Git (MUST do before adding theme!) git init # 3. Add theme git submodule add https://github.com/user/theme.git themes/theme-name # 4. Configure theme echo 'theme = "theme-name"' >> hugo.toml # 5. Test locally hugo serve -D # 6. Create content hugo new posts/my-first-post.md hugo new projects/my-project.md # 7. Commit and push git add . git commit -m "Initial commit" git branch -M main git remote add origin https://github.com/username/username.github.io.git git push -u origin main # 8. Create GitHub Actions workflow mkdir -p .github/workflows # (create hugo.yml file as shown in Part 7) # 9. Push workflow git add . git commit -m "Add GitHub Actions" git push
```

Essential Commands:

```
# Development hugo serve -D # Start dev server with drafts hugo # Build for production # Content hugo new posts/title.md # New blog post hugo new projects/name.md # New project page # Git git status # Check what changed git add . # Stage all changes git commit -m "message" # Commit changes git push # Push to GitHub
```

File Structure:

```
my-website/ ■■■ .github/ ■ ■■■ workflows/ ■ ■■■ hugo.yml # GitHub Actions config ■■■ content/ ■ ■■■ posts/ # Blog posts ■ ■ ■■■ my-post.md ■ ■■■ projects/ # Project pages ■ ■ ■■■ project-name.md ■ ■■■ about.md # Other pages ■■■ static/ ■ ■■■ images/ # Images and assets ■ ■■■ project1/ ■ ■■■ project2/ ■■■ themes/ ■ ■■■ theme-name/ # Your chosen theme ■■■ hugo.toml # Main configuration ■■■ .gitmodules # Git submodules config
```

Conclusion

Congratulations! You now have the knowledge to create and deploy a Hugo website on GitHub Pages. This setup gives you a fast, secure, and free website that you have complete control over.

Key Takeaways:

- Always run 'git init' before adding theme submodules
- Test locally with 'hugo serve -D' before pushing to GitHub
- Use draft: false when you're ready to publish content
- GitHub Actions automatically builds and deploys your site
- DNS changes can take time - be patient with custom domains
- The Hugo community is helpful - don't hesitate to ask questions

Next Steps:

- Explore different Hugo themes to find one that fits your needs
- Learn Markdown syntax for rich content formatting
- Customize your theme by editing templates and CSS
- Add analytics (Google Analytics, Plausible, etc.)
- Explore Hugo's advanced features (taxonomies, multilingual, etc.)
- Join the Hugo community forum for help and inspiration
- Consider adding a blog RSS feed for subscribers

Remember, this guide includes lessons learned from real setup experiences, including common errors and their solutions. The Hugo documentation and community forum are excellent resources whenever you get stuck. Good luck with your new website!