Build Your Personal Website

A hands-on tutorial with Jekyll and GitHub Pages

Mansur M. Arief
Industrial and Systems Engineering, KFUPM

Part 1

From zero to a live website

Tutorial Outline

Part 1: Get Something Live

- What are Jekyll and GitHub Pages?
- Create a GitHub account
- Install Git on your computer
- Open a terminal for the first time
- Create your repository on GitHub
- Add a single page and publish it
- Enable Jekyll with a theme

What Are Jekyll and GitHub Pages?

GitHub Pages is a free hosting service from GitHub. You give it a folder of files, and it serves them as a website at a URL like `yourname.github.io`.

Jekyll is a tool that turns plain text files (like Markdown) into a full website with a shared layout, navigation, and styling. You write content; Jekyll builds the HTML.

Together, they let you host a personal website by writing Markdown and pushing it to GitHub. No servers, no databases, no fees.

What you will have: a live page at `yourusername.github.io` that you control from your browser or your terminal.

Step 1: Create a GitHub Account

Go to `github.com` and click **Sign up**.

Pick your username carefully. It becomes part of your website URL.

- Username `kfupmstudent` $\Rightarrow$ website `kfupmstudent.github.io`
- Use something short, professional, and ideally your real name.
- Avoid numbers and random characters if you can.

Use a permanent email address, not your student email. You will want access to this account after you graduate.

Verify your email and you are done. You now have a GitHub account.

Step 2: Install Git on Your Computer

Git, for our purpose, is what moves files between your computer and GitHub.

On Windows: Download **Git for Windows** from `git-scm.com/download/win`. Run the installer, accept the defaults. You will get a new program called **Git Bash**.

On macOS: Open **Terminal** (press `Cmd+Space`, type `Terminal`, Enter). Then type:

```
git --version
```

If it prompts to install developer tools, accept. If it shows a version number, Git is already installed.

On Linux: Run `sudo apt install git` (Ubuntu/Debian) or `sudo dnf install git` (Fedora).

Step 3: Open a Terminal (Don't Panic)

A **terminal** is a text window where you type commands instead of clicking buttons.

Open your terminal:

- Windows: open **Git Bash** from the Start menu
- macOS: open **Terminal** from Applications / Utilities
- Linux: open **Terminal** from your applications menu

Three commands you will use over and over:

```
pwd                          (print working directory, where am I?)
ls                               (list files in the current folder)
cd foldername                    (change directory into foldername)
```

On Windows Git Bash, your Desktop lives at `/c/Users/YourName/Desktop`. On macOS and Linux, at `~/Desktop`. The `~` means "home folder."

Step 4: Tell Git Who You Are

Before Git will let you save anything, it needs your name and email. Type these commands in your terminal, replacing the quoted values with your own.

```
git config --global user.name "Your Full Name"
git config --global user.email "you@example.com"
```

Use the **same email as your GitHub account**. This links your local work to your GitHub identity.

Verify:

```
git config --global --list
```

You should see your name and email printed back.

Step 5: Create Your Website Repository

A **repository** (or *repo*) is a folder GitHub tracks for you. Your personal website lives in a very specific kind of repo.

On GitHub.com:

- Click the + icon at the top right, then **New repository**.
- For the repository name, type `yourusername.github.io` exactly. Replace `yourusername` with your actual GitHub username.
- Set it to **Public**.
- Check the box **Add a README file**.
- Click **Create repository**.

Naming rule: A repo named `kfupmstudent.github.io` (owned by user `kfupmstudent`) automatically becomes a website at that same URL.

Step 6: Download the Repository to Your Computer

You now have an empty repo on GitHub. Next, copy it to your computer so you can edit it.

In your terminal, move to your Desktop:

```
cd ~/Desktop
```

On Windows Git Bash, `~` points to your user folder.

Clone the repo (replace `yourusername` with your actual username):

```
git clone https://github.com/yourusername/yourusername.github.io.git
```

Move into the new folder:

```
cd yourusername.github.io
```

Step 7: Add Your First Page

Open the folder in text editor.

Create a new file named `index.md` inside the folder. Paste this content:

```
---
layout: default
title: Home
---

# Hello, World

I am [Your Name], a student at KFUPM.
This is my personal website.
```

The block at the top between `---` lines is called **YAML front matter**. Jekyll reads it to decide how to render the page. The rest is plain Markdown.

Step 8: Enable Jekyll with a Theme

Create another new file in the same folder named `_config.yml` (note the underscore). Paste this:

```
title: Your Name
description: Personal website
theme: minima
```

What this does:

- `title` and `description` appear in the browser tab and search results.
- `theme: minima` tells GitHub Pages to use the built-in *Minima* theme, which handles the layout, fonts, and colors for you.

That is all Jekyll needs to turn your folder into a website. No installation on your computer required because GitHub does the building for you.

Step 9: Publish Your Changes

Go back to your terminal. Make sure you are still inside the website folder (use `pwd` to check). Then run these three commands, one at a time.

```
git add .
git commit -m "Add first page and config"
git push
```

What each line does:

- `git add .` stages every changed file for saving.
- `git commit -m "..."` saves a snapshot with a message.
- `git push` uploads the snapshot to GitHub.

First push: GitHub will ask you to authenticate. A browser window opens for sign-in, and it remembers you after that.

Step 10: Visit Your Live Website

Open a browser and go to:

```
https://yourusername.github.io
```

The first build takes one to two minutes. If you see a 404 page, wait a minute and refresh.

Check the build status on GitHub: open your repository, click the **Actions** tab. A green check means your site is live. A red X means the build failed, and the log will tell you what went wrong.

You now have a live personal website. Every time you change `index.md`, commit, and push, the site updates within a minute.

The Editing Loop, From Now On

Every future change follows the same three-step rhythm.

1. **Edit a file** in your text editor and save.
2. **Commit the change** in your terminal.

```
git add .
git commit -m "short description of what changed"
```

3. **Push it up.**

```
git push
```

Wait a minute, refresh your browser, and your change is live.

Common Problems and Fixes

Symptom	Likely cause and fix
Site shows 404 after pushing	Wait 1–2 minutes for the first build. Check Actions tab.
`git push` asks for password	Use the browser sign-in prompt or a Personal Access Token.
Repo name rejected	Must be *exactly* `yourusername.github.io`, lowercase.
Page shows raw Markdown	Front matter block is missing. Check the `---` lines.
`git` command not found	Git is not installed. Revisit Step 2.
Build fails with YAML error	Use spaces, not tabs, in `_config.yml`.

When in doubt, read the error message in the **Actions** tab of your repository. It almost always names the file and line to fix.

Part 1 Checkpoint

Before moving on, you should have:

- A GitHub account with a professional username
- Git installed and configured with your name and email
- A repository named `yourusername.github.io`
- Two files in it: `index.md` and `_config.yml`
- A live site visible at `https://yourusername.github.io`

If all five are done, you are ready for Part 2, where we replace the default theme with a custom template designed for academic and professional use.

Part 2

Customizing with a Template

Part 2 Outline

A real website with a real design

- Why start from a template
- Fork our template on GitHub
- Enable GitHub Actions as the build source
- Edit `_config.yml`: name, social links, keywords
- Edit the homepage, about page, and contact page
- Add a profile photo
- Add a new page and a blog post
- Customize colors and publish

Why Start From a Template?

The site you built in Part 1 works, but looks very plain. Building a good layout from scratch would mean learning HTML, CSS, and Jekyll's theming system.

A **template** is a ready-made website you copy and customize. Someone else has already done the design work. You only change the content.

We will use the `ai-vnv/personal-website-template` repository. It is based on **Beautiful Jekyll** and includes:

- A responsive design that works on phones and laptops
- Pages for Home, About, Contact, plus an optional blog
- Support for LaTeX math, syntax-highlighted code, and comments
- Automatic deployment through GitHub Actions

Heads Up: Fresh Start

In Part 2 we replace the simple site from Part 1 with one based on the template. **The work you did in Part 1 is not wasted.** Everything you learned about Git, pushing, and the editing loop applies here too.

What we will do:

- Delete the repository you made in Part 1 (it is empty anyway).
- Fork our template repository and rename the fork to `yourusername.github.io`.
- Customize it with your information.

To delete the old repo on GitHub: open it, click **Settings**, scroll to the bottom **Danger Zone**, click **Delete this repository**, and confirm by typing the repo name.

Step 1: Fork the Template

Open the template in your browser:

```
https://github.com/ai-vnv/personal-website-template
```

Click the **Fork** button near the top right. On the fork page:

- **Owner:** your own GitHub account.
- **Repository name:** change it to `yourusername.github.io`.
- **Copy the main branch only:** leave checked.
- Click **Create fork**.

A fork is your own copy of someone else's repo. It keeps a link back to the original, but you can change anything inside your fork freely.

You now have your own copy of the template at `github.com/yourusername/yourusername.github.io`.

Step 2: Enable GitHub Actions as the Build Source

The template builds itself using **GitHub Actions** instead of GitHub's older built-in Jekyll. You tell GitHub to use Actions once, and it builds automatically on every push.

In your new repository on GitHub:

- Click **Settings** (top of the repo page).
- In the left sidebar, click **Pages**.
- Under **Build and deployment** → **Source**, select **GitHub Actions**.

That is the only setting you need. The template includes a workflow file that does the rest.

Check the **Actions** tab. You should see a build already running or finished. Once it shows a green check, your site is live.

Step 3: Download Your Fork to Your Computer

Open your terminal and move to your Desktop:

```
cd ~/Desktop
```

If the old folder from Part 1 is still on your Desktop, delete the local copy:

```
rm -rf yourusername.github.io
```

Clone *your fork* (not the original template):

```
git clone https://github.com/yourusername/yourusername.github.io.git
```

Move into it:

```
cd yourusername.github.io
```

Step 4: What's Inside the Template

Open the folder in text editor. The files that matter to you:

File or folder	What it is
`_config.yml`	Site configuration (name, social links, colors)
`index.html`	Homepage content
`about.html`	About page content
`contact.html`	Contact page content
`assets/img/`	Where your photos go
`_posts/`	Where blog posts go
`_layouts/`, `_includes/`	Template internals. You can ignore these.

Rule of thumb: folders starting with an underscore are Jekyll internals. You will only edit `_config.yml` and files in `_posts/`.

Step 5: Set Your Name in `_config.yml`

Open `_config.yml` in VS Code. Near the top you will see:

```
title:  "Your Name"
author:  "Your Name"
```

Change both to your actual name:

```
title:  "KFUPM Student"
author:  "KFUPM Student"
```

What these do:

- `title` appears in the browser tab and site header.
- `author` appears in the footer.

Save the file.

Step 6: Add Your Social Links

Scroll down in `_config.yml` to the `social-network-links` section. Every line starts with #, which means it is commented out.

Remove the # from each line you want active, and fill in your details. For example:

```
social-network-links:
email:  "kfupmstudent@example.com"
github:  kfupmstudent
linkedin:  kfupmstudent
#  twitter:  kfupmstudent      (left commented out)
```

Indentation matters in YAML. The two spaces before `email`, `github`, and `linkedin` must be spaces, not tabs. Available networks include email, github, twitter, linkedin, google-scholar, orcid, youtube, medium.

Step 7: Edit Your Homepage

Open `index.html`. At the top is a YAML front matter block, followed by content.

Replace it with your own introduction:

```
---
layout:  home
title:  KFUPM Student
subtitle:  ISE Student at KFUPM
---

Hi, I am KUFPM Student.  I study ISE
and ready to work in data science and
optimization company wherever.
```

Despite the `.html` extension, the content can be written in Markdown. The template handles the conversion.

Step 8: Edit Your About and Contact Pages

Edit `about.html` with a longer bio. Keep the front matter intact and replace only the text below the second `---`:

```
---
layout: page
title: About
subtitle: A little about me
---

I am a third-year ISE student at KFUPM
interested in supply chain optimization
and applied statistics...
```

Edit `contact.html` with your email, GitHub, and LinkedIn. The template file already has the layout; just replace the placeholder values with your own.

Step 9: Add a Profile Photo

Copy your photo into the `assets/img/` folder. A square image (around 400 by 400 pixels) works best. For example, save it as `assets/img/profile.jpg`.

In `_config.yml`, find these two commented lines:

```
# avatar:  "/assets/img/avatar-icon.png"
# round-avatar:  true
```

Uncomment both and point `avatar` to your file:

```
avatar:  "/assets/img/profile.jpg"
round-avatar:  true
```

`round-avatar:  true` crops your photo into a circle. Set it to `false` for a square.

Step 10a: Create a New Page

To add a page, create a new file e.g. `projects.md` in the root folder.

```
---
layout:  page
title:  Projects
---

### Supply Chain Simulation
Three-echelon network in SimPy.

### MLR Demand Forecast
Regression on retail sales data.
```

Save the file. Jekyll will serve it at `yourusername.github.io/projects`

Step 10b: Link the Page in the Navbar

Creating the file is not enough. You also need to add it to the navigation bar so visitors can find it.

In `_config.yml`, find the `navbar-links` section and add a new line:

```
navbar-links:
Home:  "index"
About:  "about"
Projects:  "projects"     (new line)
Contact:  "contact"
```

The label on the left (`Projects`) is what shows in the navbar. The value on the right (`"projects"`) matches the filename `projects.md` without the extension.

Step 11: Write a Blog Post

Blog posts live in `_posts/`. The filename **must** match `YYYY-MM-DD-title.md` (e.g. `2025-11-15-first-post.md`).

```
---
layout: post
title: My First Real Post
tags: [learning, statistics]
---

Write your post in Markdown. Use ## for
headings, bold, and italic.
```

The template ships with one example post. Delete it once you have added your own.

Step 12: Customize Colors (Optional)

Near the bottom of `_config.yml`, you can change the color scheme.

Default (light, blue links):

```
navbar-col:  "#EAEAEA"
link-col:   "#008AFF"
hover-col:  "#0085A1"
```

Example dark theme:

```
navbar-col:     "#1a1a2e"
navbar-text-col:  "#e0e0e0"
page-col:       "#16213e"
text-col:       "#e0e0e0"
link-col:       "#4fc3f7"
```

Step 13: Publish Your Changes

Save all files. In your terminal, from inside the repo folder:

```
git add .
git commit -m "Customize site with my info"
git push
```

This is the same editing loop from Part 1. Every change to any file in the template publishes the same way.

Visit: `https://yourusername.github.io`. If you see the old Part 1 page, hard-refresh with `Ctrl+Shift+R` (Windows/Linux) or `Cmd+Shift+R` (macOS).

Check the Actions tab on GitHub for a green check.

Template Troubleshooting

Symptom	Likely cause and fix
Build fails in Actions tab	Open the failed run. Usually a YAML indentation slip in `_config.yml`.
Profile photo doesn't show	Check the `avatar:` path. Case matters: `.JPG` $\neq$ `.jpg`.
New page not in navbar	Add it under `navbar-links`. Filename without `.md`.
Blog post not appearing	Filename must be `YYYY-MM-DD-title.md`, date today or earlier.
Social icons missing	Remove the `#` in front of the line.
Homepage still shows Part 1	Browser cache. Hard-refresh with `Ctrl+Shift+R`.

Golden rule: if a build fails, the Actions log names the file and line. Read it.

Part 2 Checkpoint

You should now have:

- A fork of the template renamed `yourusername.github.io`
- GitHub Actions enabled as the build source
- Your name, social links, and keywords in `_config.yml`
- A personalized homepage, about page, and contact page
- (Optional) a profile photo, an extra page, and a blog post
- A live site at `https://yourusername.github.io` with a proper design

What to do next: keep writing. The editing loop is always the same. Edit a file, commit, push, wait a minute, refresh. Your site grows with you.

Summary

- **GitHub Pages** gives you free hosting. **Jekyll** turns Markdown into a styled website. Together they run on any repo named `yourusername.github.io`.
- **Install once**: GitHub account, Git, a text editor. Configure Git.
- **Part 1 got you live** with two hand-written files and the `minima` theme.
- **Part 2 using a template**: forked `ai-vnv/personal-website-template`, edited `_config.yml` + other three pages, and got a live website.
- **The editing loop is the same for everything**: edit, `git add .`, `git commit -m "..."`, `git push`. Wait a minute. Refresh.
- **Your site is a long-term asset.** Keep it updated. Employers, grad schools, and collaborators will find you through it.