IMPERIAL

Early Career Researcher Institute

Introduction to Creating a Website with GitHub and Markdown

Jay DesLauriers

Imperial College London

Intended Learning Outcomes

- Create communication outputs for code or software projects *(or yourself!)*
- Format text, create tables, and insert images and code with Markdown
- Utilise Markdown syntax offered by other flavours of Markdown
- Create ~~a README, a Jupyter presentation or an R Markdown report~~

  a personal , professional or project website, or software documentation

Websites
What's in a website?

Frontend
(runs in the browser)

- HTML – structure and content
- CSS – presentation (colours, fonts, layout)
- JS (JavaScript) – behaviour and interactivity

Provides *static* content
(e.g. written copy, images)

Backend
(runs on a server)

- Database – MySQL / PostgreSQL / MongoDB
- Server logic – JS / PHP / Python / Java / C#

Handles *dynamic* features
(e.g. user login, shopping basket)

Websites
Static Site Generators (SSGs)

- Not all websites need dynamic content (or a backend)
- If a site is just frontend files, it can be hosted as a *static site*.
- Static sites are usually simpler and cheaper to host
- And Static Site Generators make static sites easy to build too:

A Static Site Generator reduces the need to write HTML, CSS and JavaScript

You write in a simpler markup language[1] and the SSG generates HTML, CSS and JS

1. Markdown is the most popular choice

Websites
...for researchers

- Personal site, showcasing your work and project experience
- Project landing page, describing a research project or group
- Blog, for personal or professional use
- Software documentation, with tutorials and how-to guides

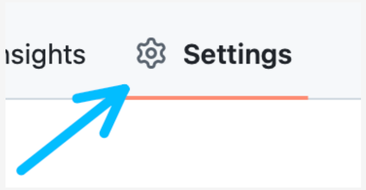
...which all work nicely as static sites !

Tech Stack
What we'll be using today

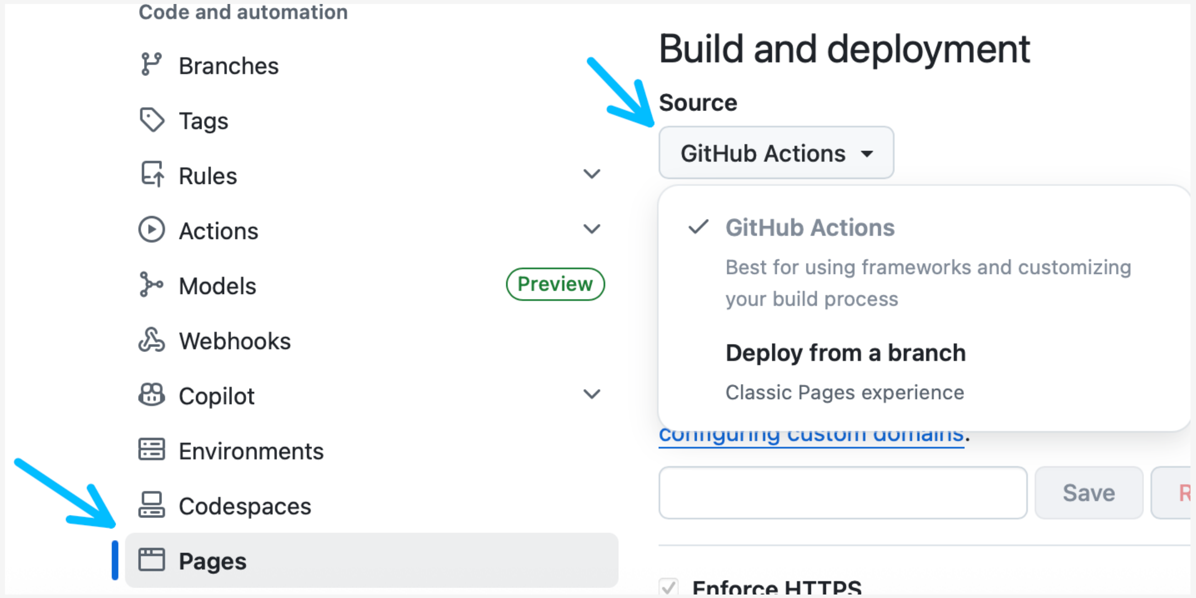
- Storing files
  - GitHub
- Markup Language
  - Markdown
- Generating HTML
  - MkDocs (Material) SSG
- Generating HTML
  - Jekyll SSG
- Automating the SSGs
  - GitHub Actions
- Hosting our site
  - GitHub Pages
- Developing our site
  - GitHub Codespaces

Appendix
Configuring Pages on your repository

1. Access your GitHub repository settings



2. Access the **Pages** menu and set **Build and deployment Source** to **GitHub Actions**



Build and deployment

Source

GitHub Actions

✓ GitHub Actions

Best for using frameworks and customizing your build process

Deploy from a branch

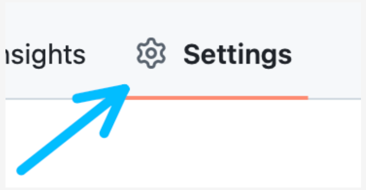
Classic Pages experience

configuring custom domains.

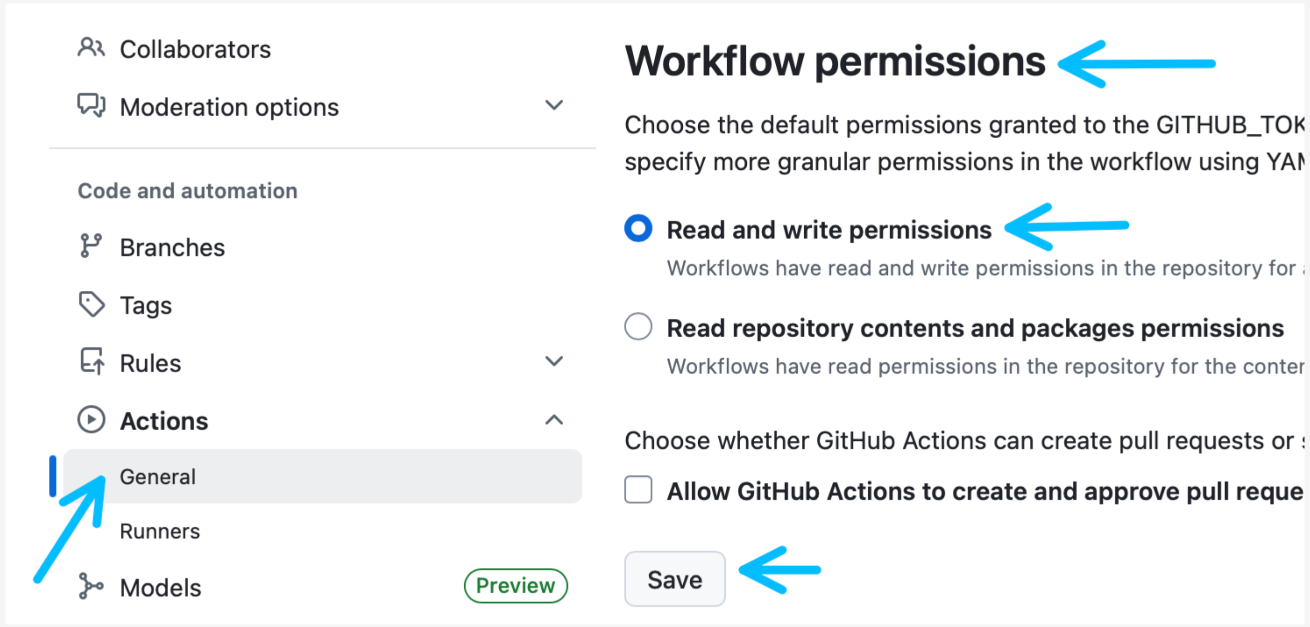
Enforce HTTPS

Appendix
Configuring Actions on your repository

1. Access your GitHub repository settings



2. Access the **Actions, General** menu; set **Workflow permissions** to **Read and write...** and **Save**



Workflow permissions

Choose the default permissions granted to the GITHUB_TOK
specify more granular permissions in the workflow using YAI

[✓ Read and write permissions]
Workflows have read and write permissions in the repository for

Read repository contents and packages permissions
Workflows have read permissions in the repository for the conter

Choose whether GitHub Actions can create pull requests or

Allow GitHub Actions to create and approve pull reque

Save

Appendix
Material for MkDocs

The quickest way to get started is to visit our template. **Use this template** to **Create a new repository**

- Then, see the appendices to configure Pages and Actions
- And follow along with the official documentation on Material for MkDocs

We created the template by combining these two resources:

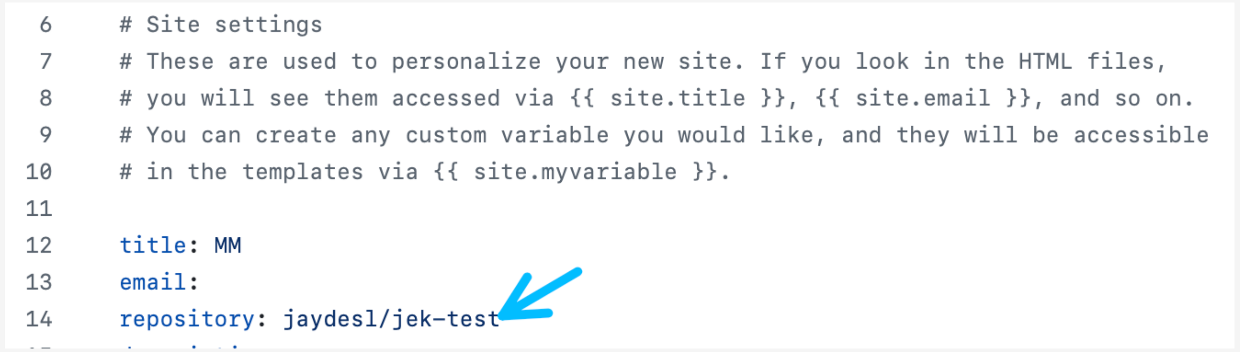
- Material for MkDocs Getting Started Guide
- GitHub Action workflow proposed here

Appendix

Jekyll

The quickest way to get started is to visit our template. **Use this template** to **Create a new repository**

- Then, see the appendices to configure Pages and Actions
- Finally, edit **_config.yml** and fill in your username and repository name next to **repository:**



```
# Site settings
# These are used to personalize your new site. If you look in the HTML files,
# you will see them accessed via {{ site.title }}, {{ site.email }}, and so on.
# You can create any custom variable you would like, and they will be accessible
# in the templates via {{ site.myvariable }}.

title: MM
email:
repository: jaydesl/jek-test
```

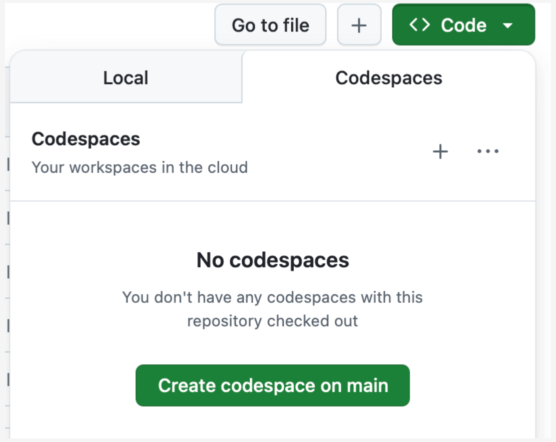
- And follow along with the official documentation on the Minimal Mistakes Jekyll theme

We created the template by combining these two resources:

- Minimal Mistakes remote theme starter
- Jekyll starter GitHub Action workflow

Appendix
Developing in Codespaces

After creating your new repository from a template,
Create a Codespace using the **Code** button:



Codespaces
Your workspaces in the cloud

No codespaces
You don't have any codespaces with this repository checked out

Wait for the Codespace to load. You can then preview your site by typing a command in the Terminal:

- For MkDocs, type `mkdocs serve --livereload`
- For Jekyll, type `bundle exec jekyll serve`

As you modify your project files in the Codespace, your preview will update to reflect those changes.
When done, click the Source Control panel on the left sidebar, type a message and click **Commit**, then **Sync**

Both templates have pre-configured Codespaces, via a devcontainer.json, which does the following
- For MkDocs, runs `pip install mkdocs-material`
- For Jekyll, runs `bundle install`

Thank you!