# mkdocs\_pages\_setup\_wizard

This interactive wizard guides you through setting up MkDocs on GitHub Pages for both the **Diplomagic\_GDD** (project) and **goldmanvision.github.io** (hub) repositories, and configuring Git in JetBrains Rider. Follow each **Step** below. After running the commands, enter **y** to confirm and proceed or **n** to repeat the step.

## Step 1: Configure GitHub Pages & Custom Domains

* **Diplomagic\_GDD (project pages):** In the GitHub repository settings (Settings → Pages), select the **gh-pages** branch (root) as the source, set the Custom domain to diplomagic.goldmanvision.com, and save. Then enable **Enforce HTTPS** (it may take a few minutes to provision). This creates a CNAME file in the gh-pages branch[[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=5,see%20%2032%20What%20is).
* **Add CNAME to MkDocs:** Ensure your project’s docs/ folder contains a file named CNAME with the line diplomagic.goldmanvision.com (so MkDocs includes it on build)[[2]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=additional%20step%20so%20that%20MkDocs,MkDocs%20will). For example:
* $ echo diplomagic.goldmanvision.com > docs/CNAME  
  $ git add docs/CNAME && git commit -m "Add custom domain CNAME"  
  $ git push
* **goldmanvision.github.io (user pages):** In its repository settings, select the **main** branch (root) as the Pages source, set the Custom domain to docs.goldmanvision.com, and enable **Enforce HTTPS**. Similarly, ensure a docs/CNAME file with docs.goldmanvision.com exists in that repo.
* **DNS Records:** In your DNS provider’s panel, create CNAME records for the subdomains, pointing them to GitHub’s domain. For example:
* Host: diplomagic Type: CNAME Value: goldmanvision.github.io  
  Host: docs Type: CNAME Value: goldmanvision.github.io
* GitHub requires subdomains to point to <user>.github.io[[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=5,see%20%2032%20What%20is).  
  *After adding these DNS records (and waiting for propagation), press* y *to confirm and continue, or* n *to review this step.*

## Step 2: Configure Git & Bash in JetBrains Rider

* **Terminal Shell:** Open Rider **Settings** (Ctrl+Alt+S) → **Tools → Terminal**. Ensure “Shell path” points to your Bash shell (e.g. /bin/bash on Linux/macOS, or bash.exe on Windows)[[3]](https://www.jetbrains.com/help/rider/Settings_Tools_Terminal.html#:~:text=Shell%20path). Rider should auto-detect a suitable shell, but you can browse to it if needed.
* **Git Executable:** In Settings → **Version Control → Git**, verify the “Path to Git executable”. Use **Browse** if necessary and then click **Test** to check it works[[4]](https://www.jetbrains.com/help/rider/Settings_Version_Control_Git.html#:~:text=Path%20to%20Git%20executable). For example, on Windows this might be C:\Program Files\Git\bin\git.exe.
* **Rider Git Workflow:** Rider offers a GUI for commits/pushes. You can enable the **staging area** option to manually stage changes before commit (Settings → Version Control → Git, enable “Use staging area”)[[5]](https://www.jetbrains.com/help/rider/Settings_Version_Control_Git.html#:~:text=Enable%20staging%20area). This mimics standard Git CLI behavior. In general, use Rider’s **Commit** dialog (or Git tool window) for commits/pushes, and use the integrated **Terminal** for running build commands like mkdocs build or deployment scripts.
* **Verify Setup:** Run:
* $ git --version && echo $SHELL
* to confirm Git is found and the shell is correct. For example, you should see something like:
* git version 2.x.x  
  /bin/bash
* If this output is correct, press y to proceed (or n to retry configuration).

## Step 3: Verify Repository Structure & Branching

* **Current Branch:** Ensure you are on the main branch in each repo. Run:
* $ git branch
* You should see \* main. If not, switch to main:
* $ git checkout main
* MkDocs recommends running deploy from the main (primary) branch[[6]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=Project%20Pages%20sites%20are%20simpler,project%2C%20run%20the%20following%20command).
* **Project Repo (Diplomagic\_GDD):** Check that a gh-pages branch exists for publishing:
* $ git branch -a
* Look for remotes/origin/gh-pages. If it doesn’t exist, it will be created on the first deploy.
* **Required Files:** Verify essential files are present in the repo root:
* $ ls .gitattributes .gitignore mkdocs.yml docs
* Each of these should exist (no “No such file” errors). The mkdocs.yml and docs/ folder are required for MkDocs, and .gitignore/.gitattributes help manage Git tracking.
* **Clean Working Directory:** Ensure no uncommitted changes:
* $ git status
* It should report clean or “nothing to commit”. If you had been on another branch, run git checkout main again.  
  *When the branch and files check out correctly, press* y *to continue (or* n *to fix issues and retry).*

## Step 4: Build & Publish **Diplomagic\_GDD** (Project Pages)

* **Navigate to project directory:**
* $ cd path/to/Diplomagic\_GDD
* **Double-check Git setup:**
* $ git remote -v  
  $ git branch
* Ensure origin points to .../Diplomagic\_GDD.git and you are on main.
* **Build the site:** Run MkDocs build:
* $ mkdocs build
* Expected output (example):
* INFO - Cleaning site directory  
  INFO - Building documentation to directory: /.../site
* After this, a site/ directory is created with the HTML.
* **Publish to GitHub Pages:** Deploy the site/ directory to the gh-pages branch. For example, using ghp-import:
* $ ghp-import -n -p -f site
* The flags mean: -n (add .nojekyll), -p (push to origin), -f (force overwrite). Alternatively, you can use:
* $ mkdocs gh-deploy --clean
* behind the scenes this uses ghp-import[[7]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=mkdocs%20gh). Expected output:
* INFO - Building documentation to directory: /.../site  
  INFO - ghp-import: Commencing import of directory '/.../site'  
  Setting up the deployment branch...  
  Switched to branch 'gh-pages'  
  [gh-pages ...] Updating files...  
  Pushing to origin gh-pages... done.
* (Any warnings about history are normal; ghp-import will overwrite the gh-pages branch as it manages it fully[[8]](https://pypi.org/project/ghp-import/#:~:text=Big%20Fat%20Warning).)
* **Return to main:** After deploying, switch back:
* $ git checkout main  
  $ git pull
* Confirm you are on main with up-to-date code.
* **Success:** If build and deploy ran without errors, press y to continue. (If there are errors or missing files, fix them and use n to retry this step.)

## Step 5: Build & Publish **goldmanvision.github.io** (User/Hub Pages)

* **Navigate to hub directory:**
* $ cd path/to/goldmanvision.github.io
* **Verify Git:**
* $ git remote -v  
  $ git branch
* Ensure origin is .../goldmanvision.github.io.git and branch is main.
* **Build the site:** Assuming this repo also contains MkDocs source:
* $ mkdocs build
* Expected output similar to Step 4, producing a site/ directory.
* **Publish to main branch:** Deploy site/ contents to the main branch. You can use ghp-import with -b main (or tell MkDocs to use main as the remote branch). For example:
* $ ghp-import -n -p -f site -b main
* (The -b main option specifies pushing to the main branch instead of gh-pages[[9]](https://pypi.org/project/ghp-import/#:~:text=bloating%20the%20repository%20size%20and,is%20highly%20recommended).) Expected output:
* INFO - Building documentation to directory: /.../site  
  INFO - ghp-import: Commencing import...  
  Setting up the deployment branch...  
  [main ...] Updating files...  
  Pushing to origin main... done.
* Alternatively, mkdocs gh-deploy --clean --remote-branch main.
* **Refresh local:**
* $ git checkout main  
  $ git pull
* **Success:** If everything pushed successfully, press y. This completes the publishing steps.

## Step 6: Validate DNS & Site Availability

* **Test the custom domains:** In the terminal, run:
* $ curl -I https://diplomagic.goldmanvision.com
* You should see an HTTP 200 response, e.g.:
* HTTP/2 200
* If you get a redirect or error, wait and re-test (DNS/HTTPS provisioning can take a few minutes).  
  Similarly test the hub domain:
* $ curl -I https://docs.goldmanvision.com
* Expect HTTP/2 200 OK (or similar).
* **Verify DNS (optional):** Ensure the DNS record is correct. On Windows (PowerShell):
* Resolve-DnsName diplomagic.goldmanvision.com
* On Linux/macOS:
* dig diplomagic.goldmanvision.com
* The answer should show a CNAME to goldmanvision.github.io.
* **Completion:** Once both domains return HTTP 200, your sites are live. Press y to finish the wizard.

Congratulations! You have configured MkDocs publishing with GitHub Pages for both repositories, verified Rider’s Git/Bash settings, and confirmed DNS/HTTPS setup.

**Sources:** Guidance is based on the MkDocs and GitHub Pages documentation[[6]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=Project%20Pages%20sites%20are%20simpler,project%2C%20run%20the%20following%20command)[[2]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=additional%20step%20so%20that%20MkDocs,MkDocs%20will)[[10]](https://pypi.org/project/ghp-import/#:~:text=This%20will%20DESTROY%20your%20%60gh,you%20will%20lose%20your%20work)[[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=5,see%20%2032%20What%20is)[[4]](https://www.jetbrains.com/help/rider/Settings_Version_Control_Git.html#:~:text=Path%20to%20Git%20executable)[[3]](https://www.jetbrains.com/help/rider/Settings_Tools_Terminal.html#:~:text=Shell%20path).

[[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=5,see%20%2032%20What%20is) Managing a custom domain for your GitHub Pages site - GitHub Docs

<https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site>

[[2]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=additional%20step%20so%20that%20MkDocs,MkDocs%20will) [[6]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=Project%20Pages%20sites%20are%20simpler,project%2C%20run%20the%20following%20command) [[7]](https://www.mkdocs.org/user-guide/deploying-your-docs/#:~:text=mkdocs%20gh) Deploying Your Docs - MkDocs

<https://www.mkdocs.org/user-guide/deploying-your-docs/>

[[3]](https://www.jetbrains.com/help/rider/Settings_Tools_Terminal.html#:~:text=Shell%20path) Terminal settings | JetBrains Rider Documentation

<https://www.jetbrains.com/help/rider/Settings_Tools_Terminal.html>

[[4]](https://www.jetbrains.com/help/rider/Settings_Version_Control_Git.html#:~:text=Path%20to%20Git%20executable) [[5]](https://www.jetbrains.com/help/rider/Settings_Version_Control_Git.html#:~:text=Enable%20staging%20area) Git | JetBrains Rider Documentation

<https://www.jetbrains.com/help/rider/Settings_Version_Control_Git.html>

[[8]](https://pypi.org/project/ghp-import/#:~:text=Big%20Fat%20Warning) [[9]](https://pypi.org/project/ghp-import/#:~:text=bloating%20the%20repository%20size%20and,is%20highly%20recommended) [[10]](https://pypi.org/project/ghp-import/#:~:text=This%20will%20DESTROY%20your%20%60gh,you%20will%20lose%20your%20work) ghp-import · PyPI

<https://pypi.org/project/ghp-import/>