

Conversación con Gemini

Exported: 12/6/2026 12:13:57

Jordi H.

Has dicho me gustaría exportar un arbol de documentación que es un repositorio de ficheros markdown en github. lo ideal sería un solo fichero pdf indexado. cómo lo hago

Gemini

Para exportar un repositorio de archivos Markdown a un único PDF profesional e indexado, la solución más robusta y alineada con la filosofía de software libre es utilizar **Pandoc** combinado con un motor de **LaTeX** (como XeLaTeX). Dado que trabajas con Python y arquitectura de sistemas, este enfoque te permitirá automatizar el proceso mediante scripts o incluso integrarlo en un pipeline de CI/

1. Preparación de los archivos

Para que el PDF mantenga un orden lógico, necesitas una lista de los archivos. Si no tienes un archivo SUMMARY.md o similar, puedes generar una lista en un archivo de texto o pasar los nombres directamente al comando. Un truco común es unir los archivos temporalmente o, mejor aún, dejar que Pandoc maneje la concatenación para respetar las referencias internas

2. Herramientas necesarias

Debes instalar los siguientes componentes en tu sistema (o usar una imagen de Docker especializada):

- Pandoc: El "navaja suiza" de la conversión de documentos.
- librsvg: Para renderizar imágenes vectoriales si las hay.
- Motor LaTeX: texlive-full o MikTeX (necesario para generar el PDF con índices y tipografía avanzada).

3. El comando de conversión

Desde la raíz de tu repositorio, puedes ejecutar un comando similar a este: Bash

CODE

```
pandoc [capitulo1].md [capitulo2].md [capitulo3].md \
-o documentacion_completa.pdf \
--from markdown \
--template my-template.tex \
--toc \
--toc-depth=3 \
--number-sections \
--pdf-engine=xelatex \
-V colorlinks=true \
-V linkcolor=blue
```

Parámetros clave:

- --toc: Genera automáticamente la Tabla de Contenidos (el índice indexado).
- --toc-depth: Define hasta qué nivel de encabezado (H1, H2, H3) llegará el índice.
- --number-sections: Enumera automáticamente los capítulos y subcapítulos.
- --pdf-engine=xelatex: Recomendado para soportar caracteres UTF-8 y fuentes modernas.

4. Automatización con Python

Si el repositorio es complejo o cambia constantemente, puedes usar un pequeño script en Python para encontrar todos los .md y lanzarlos a Pandoc:Pytho

CODE

```
import subprocess
import os

def generate_pdf():
    # Buscamos todos los md en orden alfabético o según una lógica específica
    files = sorted([f for f in os.listdir('.') if f.endswith('.md')])

    output_file = "manual_tecnico.pdf"

    cmd = [
        "pandoc",
        *files,
        "-o", output_file,
        "--toc",
        "--number-sections",
        "--pdf-engine=xelatex",
        "-V", "geometry:margin=1in",
        "-V", "lang=es-ES"
    ]

    subprocess.run(cmd)
    print(f"Documento generado: {output_file}")

if __name__ == "__main__":
    generate_pdf()
```

5. Alternativas "Out-of-the-box"

Si prefieres herramientas que ya gestionen la estructura de "árbol" de GitHub:

1. **MdBook:** Muy popular en entornos de sistemas. Está escrito en Rust, es rapidísimo y tiene un plugin (mdbook-pdf) para exportar todo el libro a un solo fichero
2. **MkDocs + Plugin PDF:** Si ya usas MkDocs para visualizar la doc, el plugin mkdocs-pdf-export-plugin generará el archivo automáticamente al compilar el sitio
3. **Eisvogel Template:** Es una plantilla de LaTeX para Pandoc que le da un aspecto extremadamente profesional y "corporativo" a los PDF sin esfuerzo adicional. Solo tienes que añadir --template eisvogel a l comando de Pando

¿El repositorio tiene una estructura de carpetas anidadas muy profunda o están todos los archivos en la raíz?

Jordi H.

Has dicho hay anidación. un par de niveles y un directorio de ilustraciones. Me gusta la idea de MdBook. dame algunas referencias para leer sobre ell

Gemini

Elegir **mdBook** es una excelente decisión, especialmente para alguien con un perfil técnico. Es la herramienta que utiliza el ecosistema de Rust para su documentación oficial; es minimalista, extremadamente rápida y s e integra perfectamente con el flujo de trabajo de Git. Para gestionar la anidación y las ilustraciones, mdBo ok utiliza un archivo central llamado SUMMARY.md, que actúa como el esqueleto del libro y el mapa para

el compilador. Aquí tienes las referencias fundamentales y los pasos clave para dominar

1. Documentación Oficial (El "Must-Read")

- The mdBook User Guide: Es la biblia del proyecto. Cubre desde la instalación hasta la configuración del archivo book.toml. Sección clave: "Structure", donde explica cómo vincular archivos en carpetas anidadas dentro del SUMMARY.m

2. Exportación a PDF (Tu requerimiento principal)

mdBook no genera PDF de forma nativa (genera HTML), por lo que necesitarás un "backend". El estándar de la comunidad es

- mdbook-pdf: Un plugin basado en Headless Chrome (vía la librería chrome-headless-render-pdf o similar) que garantiza que el PDF se vea exactamente igual que la versión web, manteniendo los hipervínculos internos e índice
- Alternativa: mdbook-pandoc. Si prefieres un control tipográfico total y ya conoces Pandoc, este plugin permite usarlo como motor de renderizado

3. Gestión de la Anidación e Imágenes

Para tu caso específico con carpetas e ilustraciones, debes estructurar tu SUMMARY.md así: Markdown

CODE

- ```
Summary

- [Introducción](intro.md)
- [Arquitectura](arquitectura/README.md)
- [Servidor](arquitectura/servidor.md)
- [Redes](arquitectura/redes.md)
- [Laboratorio](lab/config.md)
```
- Ilustraciones: Simplemente crea una carpeta /img o /assets dentro de /src. MdBook copiará estos recursos automáticamente al compilar. En Markdown los llamarás de forma relativa: ![Esquema](../img/diagrama.png)

## 4. Referencias de Configuración (`book.toml`)

Para que tu PDF sea profesional, revisa cómo configurar el archivo de parámetros:

- Configuration Reference: Aquí verás cómo cambiar el título, el autor y activar los diversos "backends" (como el de PDF)

## 5. Integración en GitHub (CI/CD)

Como tu documentación reside en GitHub, puedes automatizar la generación del PDF cada vez que hagas un push

- mdbook-publish-action: Una GitHub Action muy popular que te permite compilar el libro y, opcionalmente, subir el PDF resultante a las "Releases" o a GitHub Pages

**Consejo de arquitectura:** Si quieres mantener el espíritu de software libre y evitar dependencias pesadas, puedes ejecutar mdBook mediante un contenedor de **Podman** o **Docker**. Hay imágenes oficiales que incluyen mdbook-pdf, evitándote instalar dependencias de renderizado en tu máquina local. ¿Quieres que te ayude a esbozar un archivo book.toml inicial para tu estructura?

---

Has dicho hay alguna manera de generar la documentación desde el repo de github a cosas como readthedocs

## Gemini

¡Por supuesto! De hecho, **Read the Docs (RTD)** es el estándar de la industria para hospedar documentación de código abierto. Aunque históricamente nació muy ligado a Sphinx (el generador de Python), hoy en día soporta perfectamente **MkDocs**, que es el "primo hermano" de mdBook y quizá la mejor opción para integrarse con Read the Docs de forma nativa. Aquí tienes el mapa de ruta para conectar tu repositorio

### 1. La opción nativa: MkDocs en Read the Docs

Si te gusta la sencillez de mdBook pero quieres la infraestructura de RTD, **MkDocs** es el camino más corto. RTD tiene soporte oficial de primer nivel para él

- Configuración: Solo necesitas un archivo mkdocs.yml en la raíz.
- Anidación: Soporta perfectamente estructuras de carpetas. En el archivo de configuración defines la navegación:YAM

CODE

```
nav:
- Home: index.md
- Arquitectura:
- Servidor: arquitectura/server.md
- Redes: arquitectura/network.md
```

- PDF: Read the Docs genera automáticamente una versión PDF (y ePub) de tu documentación cada vez que actualizas el repo, sin que tengas que configurar herramientas de LaTeX en local

### 2. ¿Se puede usar mdBook en Read the Docs?

Directamente no es "plug-and-play" como MkDocs, pero se puede mediante un **workflow personalizado**. RTD permite ahora usar un archivo de configuración llamado .readthedocs.yaml donde puedes definir comandos personalizados de instalación. Podrías indicarle que descargue el binario de mdBook, compile y sirva la carpeta /book. Es un enfoque más "arquitectónico" y manual

### 3. El flujo de trabajo (CI/CD)

El proceso para integrar tu repo de GitHub con RTD es el siguiente:

1. Conexión: Te logueas en Read the Docs con tu cuenta de GitHub.
2. Importación: Seleccionas el repositorio de tu servidor/laboratorio.
3. Webhook: RTD instalará automáticamente un webhook en tu GitHub. Cada vez que hagas git push, RTD detecta el cambio, levanta una máquina virtual (un contenedor), compila tus Markdown y publica la web
4. Versiones: Puedes tener documentación para la rama main y versiones específicas para diferentes "releases" de tu laboratorio

### 4. Alternativa: GitHub Pages + GitHub Actions

Si prefieres no usar un servicio externo como RTD y mantenerlo todo dentro de GitHub:

- Usas una GitHub Action que ejecute mdBook.
- El PDF y el HTML se generan en el pipeline.
- El HTML se despliega en GitHub Pages.
- El PDF se adjunta como un "artifact" o se sube a una carpeta del mismo sitio.

## Resumen de herramientas para tu perfil:

| Característica | MkDocs + Read the Docs     | mdBook + GitHub Actions         |
|----------------|----------------------------|---------------------------------|
| Facilidad PDF  | Automática (nativa en RTD) | Requiere plugin y motor externo |
| Lenguaje       | Python (muy afín a ti)     | Rust (binario único)            |
| Hosting        | readthedocs.io             | github.io                       |
| Estética       | Muy profesional/limpia     | Estilo "Libro técnico"          |

**Mi recomendación:** Dado que ya trabajas con **Python** y buscas algo que gestione bien el "árbol" de ficheros y genere el PDF sin complicaciones, **MkDocs con el tema Material** desplegado en Read the Docs es la combinación más potente y con menos fricción. ¿Te gustaría que viéramos cómo estructurar el archivo `.readthedocs.yaml` para automatizarlo?

### Jordi H.

Has dicho vamos a abrir más las opciones. un generador de web estática desde el repositorio? en github hay algo ya preprado

### Gemini

Si buscamos ampliar el horizonte hacia los **Generadores de Sitios Estáticos (SSG)** que se integren de forma nativa con GitHub, entramos en el terreno donde la automatización es la norma. GitHub tiene una integración profunda con estas herramientas a través de **GitHub Pages**. Aquí tienes las opciones más sólidas y "preparadas" para lo que buscas

## 1. Jekyll (El nativo de GitHub)

Es la opción por defecto. GitHub Pages nació con Jekyll integrado, lo que significa que puedes subir tus archivos Markdown y GitHub los compilará automáticamente sin que tú hagas nada

- Ventaja: No necesitas configurar Actions si usas los temas básicos. Es "subir y listo".
- Contras: Está basado en Ruby, lo cual puede resultarte ajeno si prefieres Python. Además, generar un PDF desde Jekyll no es nativo; requiere plugins adicionales o herramientas externas

## 2. Docusaurus (El estándar moderno)

Desarrollado por Meta (Facebook), es probablemente el SSG de documentación más potente actualmente. Está basado en **React**

- Estructura de árbol: Es excelente gestionando anidaciones profundas, barras laterales dinámicas y versiones de documentos
- Búsqueda: Incluye integración con Algolia para tener un buscador profesional en tu doc.
- Despliegue: GitHub tiene una "Action" oficial para Docusaurus que despliega a GitHub Pages en segundos.

## 3. MkDocs + Material Theme (La recomendación para tu perfil)

Siendo tú usuario de **Python**, esta es la opción ganadora. **Material for MkDocs** es, posiblemente, el tema de documentación más bonito y funcional del ecosistema Open Source

- Plugin de PDF: Existe el plugin `mkdocs-pdf-export-plugin` o `mkdocs-with-pdf`.
- Directorio de ilustraciones: Gestiona las imágenes de forma impecable y permite usar Mermaid.js para crear diagramas (flujos de datos, arquitectura) directamente con texto Markdown

- GitHub Actions: Hay una acción específica (mkdocs-gh-deploy) que hace todo el trabajo por ti.

## ¿Cómo se configura en GitHub?

GitHub tiene una funcionalidad llamada **GitHub Actions**. En lugar de compilar tú el sitio en local, dejas un pequeño archivo de configuración (YAML) en la carpeta `.github/workflows/`. Cuando haces un git push

1. Una máquina virtual de GitHub se enciende.
2. Instala el generador (MkDocs, mdBook o Jekyll).
3. Compila tus Markdown.
4. Genera el PDF.
5. Publica la web en tu URL de github.io y deja el PDF disponible para descarga.

## Comparativa de flujo de trabajo

| Herramienta | Dificultad | Lenguaje Base    | Ideal para...          |
|-------------|------------|------------------|------------------------|
| Jekyll      | Baja       | Ruby             | Proyectos simples y    |
| MkDocs      | Media      | Python           | Documentación técnica, |
| Docusaurus  | Media-Alta | JavaScript/React | Sitios web de          |
| mdBook      | Baja       | Rust             | Manuales técnicos tipo |

## Mi sugerencia técnica

Dado que estás montando un **homelab** y un **vector tecnológico**, **MkDocs con el tema Material** te dará el equilibrio perfecto

1. Es software libre.
2. Usa Python (tu lenguaje principal).
3. Permite insertar diagramas de arquitectura como código.
4. La comunidad es enorme y hay cientos de extensiones.

¿Te gustaría que te pase un ejemplo de cómo sería el archivo de flujo de trabajo para que GitHub haga todo este proceso de "Markdown a Web + PDF" automáticamente

**Jordi H.**

Has dicho puedo generar la docu a docusaurus sin prácticamente interacción?

**Gemini**

Sí, es totalmente posible. De hecho, el ecosistema de **Docusaurus** está diseñado para que, una vez configurada la infraestructura inicial, tu única interacción sea hacer un git push de tus archivos Markdown. Para lograr esa "interacción cero", el secreto está en delegar todo el trabajo pesado a **GitHub Actions**. Aquí te detallo cómo automatizarlo para que el sistema se encargue de la estructura de árbol, las imágenes y el despliegue

### 1. El "Truco" de la Estructura Automática

Docusaurus suele requerir que definas un archivo `sidebars.js`. Sin embargo, para evitar interactuar con él, puedes configurar el **autogenerador de barras laterales**. En tu archivo `sidebars.js`, simplemente pon esto: `JavaScript`

CODE

```
module.exports = {
 mySidebar: [
 {
 type: 'autogenerated',
 dirName: '.', // Genera el árbol basado en tu estructura de carpetas
 },
],
};
```

Con esto, si creas una carpeta /redes con tres .md dentro, Docusaurus creará automáticamente la sección y el índice en la web sin que tú escribas una sola línea de código adicional

## 2. Automatización Total con GitHub Actions

Para no tener que instalar nada en tu servidor o PC, creas un archivo en .github/workflows/deploy.yml. Este script hará lo siguiente cada vez que detecte cambios

1. Levanta un contenedor con Node.js.
2. Instala Docusaurus.
3. Busca tus imágenes en el directorio de ilustraciones que mencionaste.
4. Compila el sitio estático.
5. Lo publica en GitHub Pages.

### Ejemplo de Workflow (YAML):

YAML

CODE

```
name: Deploy Docusaurus
on:
 push:
 branches: [main]
jobs:
 deploy:
 runs-on: ubuntu-latest
 steps:
 - uses: actions/checkout@v4
 - uses: actions/setup-node@v4
 with:
 node-version: 18
 - name: Install dependencies
 run: npm install
 - name: Build website
 run: npm run build
 - name: Deploy to GitHub Pages
 uses: peaceiris/actions-gh-pages@v3
 with:
 github_token: ${ secrets.GITHUB_TOKEN }
 publish_dir: ./build
```

## 3. Manejo de Ilustraciones

Como mencionaste que tienes un directorio de ilustraciones, Docusaurus las gestiona de forma nativa. Si tus imágenes están en /static/img/, puedes referenciarlas en tus Markdown así  
![Arquitectura del Lab](/img/esquema.png)Docusaurus se encarga de optimizarlas y servir las correctamente en la web final

## 4. ¿Y el PDF único?

Aquí es donde Docusaurus requiere un pequeño paso extra. Como es un generador orientado a web (Single Page Application), no genera PDF nativo al vuelo. Tienes dos opciones para mantener la "baja interacción"

- Plugin docusaurus-plugin-content-docs: Permite añadir un botón de "Descargar PDF" que utiliza servicios como PrinceXML o DocRaptor
- Action de terceros: Puedes añadir un paso final en tu GitHub Action que use Wandoc o un script de Puppeteer para "escanear" tu sitio ya publicado y convertirlo en un PDF indexado de forma automática

### Resumen: ¿Cómo queda tu flujo de trabajo?

1. Escribes tus notas sobre el homelab en un .md dentro de tu repo.
  2. Haces git commit y git push.
  3. Fin.
- GitHub Actions compila la web.
  - La web se actualiza en tu URL de GitHub.
  - El árbol de navegación se ajusta solo si añadiste carpetas.

Es lo más parecido a "magia" documental que puedes conseguir manteniendo el control total del código.

¿Quieres que te ayude a preparar la estructura inicial de carpetas para que Docusaurus la reconozca a la primera?

Jordi H.

Has dicho si

Gemini

Para poner en marcha **Docusaurus** con el mínimo esfuerzo y que se gestione solo, la estructura de tu repositorio debería verse así. Esto permitirá que el autogenerador de barras laterales (el "árbol") y el directorio de ilustraciones funcionen sin que tengas que tocar código cada vez que añades contenido.

### 1. Estructura recomendada del repositorio

Docusaurus espera una disposición específica para encontrar las cosas automáticamente: Plaintext

CODE

```
mi-repo-documentacion/
docs/ <-- Aquí van tus .md (puedes anidar carpetas aquí)
intro.md
infraestructura/
servidor.md
redes.md
laboratorio/
herramientas.md
static/ <-- Directorio de ilustraciones
img/
esquema-red.png
docusaurus.config.js <-- Configuración principal
sidebars.js <-- Aquí activamos el "auto-tree"
package.json
.github/workflows/ <-- El motor de automatización
```

### 2. Configuración del "Árbol" automático

En tu archivo `sidebars.js`, olvida listar archivo por archivo. Usa esto para que Docusaurus escanee tu carpeta `docs/` y cree el índice por tí:JavaScript

CODE

```
module.exports = {
 tutorialSidebar: [{type: 'autogenerated', dirName: '.'}],
};
```

### 3. Configuración del Proyecto (`docsaurus.config.js`)

Para que las ilustraciones funcionen sin problemas, asegúrate de que el `baseUrl` y el `url` apunten a tu dirección de GitHub Pages:JavaScript

CODE

```
module.exports = {
 title: 'Mi Homelab',
 url: 'https://tu-usuario.github.io',
 baseUrl: '/tu-repo/',
 onBrokenLinks: 'throw',
 onBrokenMarkdownLinks: 'warn',
 organizationName: 'tu-usuario',
 projectName: 'tu-repo',
 // ... resto de la config
};
```

### 4. Generación del PDF (Interacción Cero)

Como Docusaurus no genera el PDF por defecto, lo más eficiente para tu perfil es añadir un paso extra en la **GitHub Action**. Podemos usar una herramienta llamada `pagedjs` o un plugin específico que convierta el sitio web ya generado en un único PDF. Si quieres el PDF "indexado y profesional" que mencionabas al principio, una opción muy limpia es usar un contenedor de **Playwright** en el workflow que "navegue" por tu documentación y la imprima a PDF después de compilar

#### Pasos para empezar hoy mismo:

1. Inicializa el repo: Si tienes Node.js, ejecuta `npx create-docusaurus@latest my-website classic`.
2. Mueve tus Markdown: Pon tus archivos actuales en la carpeta `docs/`.
3. Configura el Workflow: Crea el archivo `.github/workflows/deploy.yml` que te pasé antes.
4. Push: En cuanto hagas el push, GitHub detectará todo y en un par de minutos tendrás tu web de documentación operativa

¿Prefieres que te ayude a detallar el paso exacto del Workflow para generar el PDF de forma automática al

Jordi H.

Has dicho puedo generar acceso privado a la web resultante? con user/pass?

Gemini

Al tratarse de una web estática servida por **GitHub Pages**, no existe una base de datos ni un servidor de aplicaciones que gestione sesiones de usuario de forma nativa. Sin embargo, existen tres caminos principales para lograr esa privacidad sin perder la automatización

## 1. El método "Nativo" (GitHub Pro / Enterprise)

Si tu repositorio es privado y tienes una cuenta de **GitHub Pro, Team o Enterprise**, GitHub Pages ofrece una opción llamada **"Access Control"**

- **Cómo funciona:** En la configuración del repositorio, puedes restringir la visualización de la web solo a personas que tengan acceso al repositorio
- **Pros:** Es oficial, seguro y gestionado por GitHub.
- **Contras:** Requiere suscripción de pago y que los usuarios tengan cuenta en GitHub.

## 2. Cloudflare Pages (La mejor opción gratuita y profesional)

Si buscas algo profesional con usuario y contraseña sin pagar licencias de GitHub, la solución ideal es conectar tu repo a **Cloudflare Pages**

- **Cómo funciona:** Cloudflare importa tu código de GitHub y lo sirve. Puedes activar Cloudflare Access (parte de su plan "Zero Trust", gratuito hasta 50 usuarios)
- **Privacidad:** Permite configurar un formulario de login antes de ver la web. Puedes usar contraseñas temporales por email (OTP) o integrar proveedores como Google, GitHub o incluso un simple usuario/contraseña
- **Sincronización:** Cada vez que hagas un push en GitHub, Cloudflare recompila el Docusaurus automáticamente

## 3. Stork / Static Crypt (Protección por contraseña en el cliente)

Si quieres mantenerlo en GitHub Pages gratuito y solo necesitas una barrera simple (sin gestión compleja de usuarios), puedes cifrar el contenido estático

- **Herramienta:** StaticCrypt.
- **Cómo funciona:** Se añade un paso al final de tu GitHub Action que cifra tus archivos HTML con una contraseña
- **Experiencia:** Cuando alguien entra en la web, ve una caja de texto que pide la contraseña. Al introducirla, el navegador descifra el contenido en local
- **Pros:** Muy fácil de implementar y 100% gratuito.
- **Contras:** Si alguien conoce la contraseña, puede compartirla. No es una seguridad de nivel bancario, pero es un "candado" muy efectivo para documentaciones personales o de laboratorio

## Mi recomendación técnica

Como ya gestionas un **homelab** y te gusta el software libre, yo optaría por **Cloudflare Pages + Cloudflare**

1. **Seguridad Real:** No solo ocultas la web, sino que está protegida por una capa de red global.
2. **Facilidad:** El flujo de CI/CD es idéntico: tú solo haces push a GitHub y Cloudflare se encarga del resto.
3. **PDF Privado:** Puedes dejar el link de descarga del PDF dentro de la web protegida, asegurando que solo los autorizados bajen tu documentación completa

¿Te interesa que profundicemos en cómo configurar el **StaticCrypt** dentro de tu GitHub Action para que sea totalmente automático? Es la opción que requiere menos cambios de infraestructura

**Jordi H.**

Has dicho puedo usar uno de los repositorios como "central" (de hecho, solamente contiene ficheros markdown e ilustraciones), y además incorporar las documentaciones de otros repositorios de código fuente ? muchos de mis repos tienen una sección docs con el informe técnico, de uso, u otras informaciones

Es una arquitectura excelente y muy común en entornos profesionales: se conoce como **Documentación n Publicación Centralizada**\*\*. Dado que tu repositorio central será el "cerebro" (el que tiene la estructura de Docusaurus/MkDocs) y los demás repositorios contienen la información técnica específica, tienes dos formas principales de unificar todo sin tener que copiar y pegar archivos manualmente.

1. La opción nativa de Git: Git Submodules

Es la solución más pura desde el punto de vista de la arquitectura de sistemas. Consiste en "anclar" otros repositorios dentro de una carpeta de tu repositorio central

- **Cómo funciona:** En tu repo central, ejecutas `git submodule add [URL-repo-fuente] docs/proyecto-A`.
- **Automatización:** Cuando tu GitHub Action vaya a compilar la web, le indicas que primero "traiga" el contenido de los submódulos
- **Sincronización:** El repo central siempre apuntará a una versión (commit) específica del repo de código. Si actualizas el informe técnico en el repo de código, solo tienes que hacer un `git submodule update` en el central para que la web se actualice

2. El enfoque de "Multi-repo" con GitHub Actions (Recomendado)

Esta es la opción más flexible y la que mejor encaja con la idea de "interacción cero". En lugar de que el repo central "tire" de los demás, configuramos el flujo para que los datos "fluyan" hacia el centro. **El flujo sería este**

1. Tienes tu Repo Central con la estructura de Docusaurus.
2. En tus Repos de Código, configuras una GitHub Action que, cada vez que edites algo en la carpeta /docs, haga un "push" automático solo de esos archivos hacia una carpeta específica del Repo Central
3. El Repo Central, al recibir esos archivos nuevos, dispara su propia compilación y actualiza la web.

3. Usar Plugins específicos (MkDocs "Remote" / Docusaurus "Remote Content")

Existen herramientas diseñadas específicamente para este "puzzle":

- **MkDocs Remote Content:** Existe un plugin llamado `mkdocs-remote-content` que te permite definir en el archivo de configuración URLs de otros repositorios. Al compilar, MkDocs descarga los Markdown de esos sitios externos y los renderiza como si fueran locales
- **Docusaurus Remote Content:** Hay plugins similares (como `docusaurus-plugin-remote-content`) que descargan los archivos durante el proceso de build

Comparativa para tu escenario

| Característica | Git Submodules                | GitHub Actions (Push)      | Plugins Remote Content    |
|----------------|-------------------------------|----------------------------|---------------------------|
| Control        | Total (tú eliges qué versión) | Automático al 100%         | Dinámico al compilar      |
| Complejidad    | Media (Git puede ser          | Alta al inicio (configurar | Baja (solo configuración) |
| Uso de Red     | Solo al clonar                | Frecuente (cada push)      | En cada compilación       |

Mi recomendación para tu Homelab

Si quieres que el mantenimiento sea mínimo, usa los **Plugins de Remote Content**. Simplemente le dices a tu Docusaurus central: "Busca el fichero README.md de este otro repositorio e insértalo en la sección 'Servidores'". De esta forma, mantienes el código y su documentación juntos (como debe ser), pero el usuario final ve una web única, coherente y perfectamente indexada. ¿Te gustaría ver cómo se configuraría una sección de "Remote Content" en el fichero de Docusaurus para traer un archivo de otro re

Jordi H.

Has dicho ya tengo funcionando una primera aproximacion. muy bien el docusaurus. ahora quiero que la url no sea de esas inventadas por github. que la pueda decidir yo o que tenga otra opcio

## Gemini

¡Excelente! Me alegra que **Docusaurus** ya esté corriendo. Ahora, para personalizar la URL y profesionalizar el acceso, tienes dos caminos principales según si ya posees un dominio propio o si buscas una opción gratuita a con mejor estética. Aquí tienes las opciones para dejar atrás la URL por defecto (`usuario.github.io/repo/`)

### 1. Usar un Dominio Propio (Custom Domain)

Si tienes un dominio (ej. `mi-laboratorio.com` o `hernan.dev`), puedes vincularlo directamente a tu documentación en GitHub

- Configuración en GitHub: En tu repositorio, ve a Settings > Pages. En la sección "Custom domain", escribe tu dominio o subdominio (ej. `docs.mi-laboratorio.com`)
- Configuración DNS: Deberás ir al panel de tu registrador de dominios (donde lo compraste) y añadir: Un registro CNAME apuntando a `tu-usuario.github.io`. O registros A apuntando a las IPs oficiales de GitHub si usas el dominio raíz
- SSL/HTTPS: GitHub generará automáticamente el certificado SSL (Let's Encrypt) para tu dominio de forma gratuita

### 2. Cloudflare Pages (Recomendado por estética y privacidad)

Como mencionamos antes, si mueves el despliegue a **Cloudflare Pages** (que se conecta a tu repo de GitHub con un clic)

- URL más limpia: Te darán una URL del tipo `mi-docs.pages.dev`, que suele ser más corta y profesional que la de GitHub
- Dominio personalizado gratis: Cloudflare facilita enormemente la gestión del SSL y la seguridad.
- Ventaja extra: Desde aquí puedes activar el Acceso Privado (login con email o clave) de forma mucho más robusta que en GitHub Pages

### 3. Cambiar el "BaseURL" de GitHub Pages

Si no quieres comprar un dominio pero quieres que la URL sea más corta dentro de GitHub, puedes configurar el repositorio como tu "User Site"

- Si renombras tu repositorio a `tu-usuario.github.io`, la documentación pasará de estar en `tu-usuario.github.io/proyecto/` a estar directamente en la raíz  
: `[https://tu-usuario.github.io](https://tu-usuario.github.io)`

## Ajuste CRÍTICO en Docusaurus

Independientemente de la opción que elijas, **debes actualizar tu archivo `docusaurus.config.js`**, de lo contrario los estilos CSS y las imágenes se romperán al cambiar la URL:JavaScript

CODE

```
// docusaurus.config.js
module.exports = {
 url: 'https://tu-nuevo-dominio.com', // O 'https://mi-docs.pages.dev'
 baseUrl: '/', // Si ahora está en la raíz del dominio, cámbialo de '/repo/' a '/'
 // ...
};
```

## Resumen de pasos para el cambio:

1. Decide el nombre: ¿Comprarás un .com/.es o prefieres un subdominio gratuito de Cloudflare?
  2. Configura el Hosting: En los Settings de GitHub o el panel de Cloudflare.
  3. Actualiza el código: Cambia url y baseUrl en tu docusaurus.config.js y haz push.
- Si optas por un **dominio propio**, ¿necesitas que te ayude con los registros DNS específicos que debes crear?

### Jordi H.

Has dicho estoy dentro de una organization. cada repo genera una url inventada con palabras aleatorias. a mi me ha tocado effective-adventure-xxxxxx.page

### Gemini

Parece que GitHub ha activado por defecto el aislamiento de subdominios para tu organización, lo que genera esos nombres aleatorios por seguridad (para evitar ataques de tipo cross-cookie entre diferentes proyectos de la misma organización). Como estás dentro de una **GitHub Organization**, tienes tres niveles para solucionar esto y conseguir una URL que no parezca generada por un generador de contraseñas

#### 1. El método "Limpio": Custom Domain (Recomendado)

Si tu organización ya tiene un dominio corporativo (ej. empresa.com), puedes crear un subdominio específico para tu documentación (ej. docs-laboratorio.empresa.com)

1. En GitHub: Ve a los Settings de la Organización > Pages. Allí puedes añadir el dominio raíz de la empresa y verificarlo
2. En el Repo: Ve a Settings > Pages del repositorio de Docusaurus y escribe el subdominio que quieras usar.
3. DNS: Pide al administrador de sistemas que apunte ese CNAME a tu-org.github.io.

#### 2. Renombrar el Repositorio (Si usas el dominio de la Org)

Si no tienes un dominio propio, GitHub Pages usa por defecto la estructura:

`https://<org-name>.github.io/<repo-name>/`. Si tu URL actual es algo como

`effective-adventure-xxxxxx.pages.github.io`, es probable que la organización esté usando **GitHub Enterprise** con subdominios únicos por repositorio. Para cambiar el nombre "adventure", simplemente

1. Ve a Settings > General.
2. Cambia el Repository Name por algo descriptivo como doc-homelab.
3. La URL debería actualizarse automáticamente a algo tipo `doc-homelab-xxxxxx.pages.github.io`. Sigue teniendo el sufijo aleatorio por seguridad, pero al menos el cuerpo de la URL es legible

#### 3. La opción "Pro": Cloudflare Pages

Dado que mencionaste que te gusta tener control y eres arquitecto de sistemas, **Cloudflare Pages** es la solución más elegante para saltarse las restricciones de nombres de las organizaciones de GitHub

1. Conectas Cloudflare a tu repositorio de la organización.
2. Cloudflare te permite elegir el nombre del proyecto. Si eliges mi-laboratorio, tu URL será `[https://mi-laboratorio.pages.dev](https://mi-laboratorio.pages.dev)`
3. Es gratuito, mucho más rápido que GitHub Pages y elimina por completo las palabras aleatorias.

### ¡Ojo con el archivo de configuración!

En el momento en que cambies esa URL "inventada" por una propia o una de Cloudflare, **Docusaurus dejará de**

si no actualizas el docusaurus.config.js. Asegúrate de cambiar estos valores para que coincidan con tu nueva identidad: JavaScript

CODE

```
// docusaurus.config.js
module.exports = {
 url: 'https://tu-nueva-url.com', // La nueva URL (sin el path)
 baseUrl: '/', // Si la nueva URL apunta directamente a la raíz de la doc
 // ...
};
```

¿Tienes permisos de administrador en la organización o prefieres una solución que puedas gestionar tú solo sin

---