n8n + GitHub (GRÁTIS): upload de PDF (binário) e retornar URL pública

Este guia te leva do zero até ter um **fluxo n8n** que recebe um **arquivo binário (PDF)**, envia para um **repositório GitHub** e devolve uma **URL pública** do arquivo.

Você terá **duas URLs** ao final:

- **Raw (instantânea):** `https://raw.githubusercontent.com/<usuario>/<repo>/<branch>/<caminho/arquivo.pdf>`
- **GitHub Pages (bonita/opcional):** `https://<usuario>.github.io/<repo>/<caminho/arquivo.pdf>`

  Observação: links no GitHub são **HTTPS** (seguro). Se quiser usar **domínio próprio** depois, dá para apontar um **CNAME** no GitHub Pages.

---

0) Pré-requisitos

- Conta gratuita no **GitHub** (github.com)
- Acesso ao **n8n**

---

1) Criar o repositório no GitHub

1. Acesse o GitHub > clique em **New** (novo repositório).
2. Nome sugerido: `arquivos-publicos` (pode ser outro).
3. **Public** (público).
4. Clique **Create repository**.

   Depois vamos jogar os PDFs numa pasta, ex.: `uploads/`.

---

2) Criar um token de acesso (PAT)

O n8n vai autenticar na API do GitHub usando um **Personal Access Token**. Recomendo **fine-grained token** (escopos restritos a um repo):

1. GitHub (logado) > **Settings** (Configurações) > **Developer settings** > **Personal access tokens** > **Fine-grained tokens** > **Generate new token**.
2. **Resource owner**: seu usuário.
3. **Repository access**: **Only selected repositories** > selecione **seu repositório** (ex.: `arquivos-publicos`).
4. **Permissions**:
5. **Repository permissions** > **Contents**: **Read and Write**
6. **Repository permissions** > **Metadata**: **Read**
7. Clique **Generate token** e **copie** o token (guarde em lugar seguro; você não verá de novo).

Alternativa (mais simples, porém ampla): **token clássico** com escopo `public_repo` (ou `repo` se o repo for privado).

Referências oficiais: API de conteúdo de arquivos e tokens.

---

3) (Opcional) Ativar GitHub Pages para ter URL "bonita"

Se quiser também a URL no formato `https://<usuario>.github.io/<repo>/...`:

1. No repositório > **Settings** > menu **Pages**.
2. Em **Build and deployment** > **Source**: **Deploy from a branch**.
3. **Branch**: `main` e **/ (root)** (ou `docs/` se preferir).
4. **Save**. Aguarde 1–3 minutos no primeiro publish.

A partir daí, qualquer arquivo commitado nesse branch/folder fica acessível por `https://<usuario>.github.io/<repo>/<caminho/arquivo.pdf>`

Referências oficiais: Quickstart e configuração do GitHub Pages.

---

4) Fluxo n8n (sem código) usando HTTP Request

A ideia: ler um PDF binário e fazer um `PUT` na API **Create/Update file contents** do GitHub.

4.1) Nós do fluxo

1. **Manual Trigger** (para testar)
2. **Read Binary File** (opcional, só para teste)
3. **File Path**: um PDF local (ex.: `/data/teste.pdf`)
4. **Binary Property**: `data`
5. **Set** (metadados)
6. *Keep Only Set*: **false** (para não perder o binário)
7. Campos JSON (adicione):
   - `owner`: `SEU_USUARIO_GITHUB`
   - `repo`: `arquivos-publicos`
   - `branch`: `main`
   - `folder`: `uploads` (ou vazio para raiz)
   - `filename`: `arquivo.pdf` (ou dinâmico de outro nó)
8. **HTTP Request** (o upload)
9. **Method**: `PUT`
10. **URL**:

```
https://api.github.com/repos/{{ $json.owner }}/{{ $json.repo }}/
contents/{{ $json.folder ? $json.folder + '/' : '' }}
{{ $json.filename }}
```

11. **Authentication**: None (vamos mandar header manual)

12. **Headers**:
    - `Authorization` : `Bearer SEU_TOKEN_AQUI`
    - `Accept` : `application/vnd.github+json`
    - `X-GitHub-Api-Version` : `2022-11-28`
13. **Send Body As**: **JSON**
14. **Body Parameters** (JSON):
    - `message` → `{{$json.filename}} via n8n`
    - `content` → `{{$binary.data.data}}` *(o PDF **já** está em base64)*
    - `branch` → `{{$json.branch}}`
15. **Response Format**: JSON

    Dica: se seu binário não se chama `data`, troque em `{{$binary.data.data}}` para o nome correto (ex.: `{{$binary.file.data}}`).

1. **Set** (construir URLs de retorno)
2. *Keep Only Set*: **true**
3. Campos JSON (com **Expressions**):
    - `path` → `{{$json.content.path}}`
    - `rawUrl` → `https://raw.githubusercontent.com/{{$item(0).$node["Set"].json.owner}}/{{$item(0).$node["Set"].json.repo}}/{{$item(0).$node["Set"].json.branch}}/{{$json.content.path}}`
    - `pagesUrl` → `https://{{$item(0).$node["Set"].json.owner}}.github.io/{{$item(0).$node["Set"].json.repo}}/{{$json.content.path}}`

Pronto! A saída desse **Set** final terá algo como:

```
{
  "path": "uploads/arquivo.pdf",
  "rawUrl": "https://raw.githubusercontent.com/SEU_USUARIO/arquivos-publicos/main/uploads/arquivo.pdf",
  "pagesUrl": "https://SEU_USUARIO.github.io/arquivos-publicos/uploads/arquivo.pdf"
}
```

> **Raw** funciona imediatamente após o commit. **Pages** funciona depois que você ativar o GitHub Pages (passo 3) e pode levar alguns minutos na primeira vez.

4.2) Evitar erro ao sobrescrever (422)

Se você tentar subir **o mesmo caminho/arquivo** de novo, a API exige o campo `sha` (do arquivo existente). Simples:

- Estratégia fácil: **nomeie com timestamp** (ex.: `arquivo-{{$now}}.pdf`).
- Estratégia "sobrescrever" (avançada):
- **HTTP Request (GET)** `https://api.github.com/repos/{{owner}}/{{repo}}/contents/{{folder}}/{{filename}}?ref={{branch}}` \ Se 200, pegue `sha`.
- **HTTP Request (PUT)** igual ao upload, **incluindo** `sha` no body:

```
{
  "message": "update via n8n",
  "content": "<base64>",
  "branch": "main",
  "sha": "<sha_do_arquivo_existente>"
}
```

---

5) (Opcional) Mesma coisa via Code Node (JavaScript)

Cole no nó **Code** (Run Once for Each Item):

```
const item = $input.item;
const binary = item.binary.data; // troque se a propriedade tiver outro nome

const owner = 'SEU_USUARIO_GITHUB';
const repo = 'arquivos-publicos';
const branch = 'main';
const filePath = `uploads/${item.json.filename || 'arquivo.pdf'}`; // evite
espaços no nome

const token = 'SEU_TOKEN_AQUI'; // ou use credenciais/variáveis de ambiente

let sha = null;
// (opcional) detectar se já existe para permitir overwrite
try {
  const exists = await this.helpers.httpRequest({
    method: 'GET',
    url: `https://api.github.com/repos/${owner}/${repo}/contents/$
{encodeURIComponent(filePath)}?ref=${branch}`,
    headers: {
      Authorization: `Bearer ${token}`,
      Accept: 'application/vnd.github+json',
      'X-GitHub-Api-Version': '2022-11-28',
    },
    json: true,
  });
  sha = exists.sha || exists.content?.sha;
} catch (e) {
  // 404 = arquivo não existe, segue sem sha
}

const body = {
  message: `upload via n8n: ${item.json.filename || 'arquivo.pdf'}`,
  content: binary.data, // já é base64 no n8n
  branch,
};
if (sha) body.sha = sha;
```

```
const res = await this.helpers.httpRequest({
  method: 'PUT',
  url: `https://api.github.com/repos/${owner}/${repo}/contents/$
{encodeURIComponent(filePath)}`,
  headers: {
    Authorization: `Bearer ${token}`,
    Accept: 'application/vnd.github+json',
    'X-GitHub-Api-Version': '2022-11-28',
  },
  body,
  json: true,
});

return {
  json: {
    path: res.content.path,
    rawUrl: `https://raw.githubusercontent.com/${owner}/${repo}/${branch}/$
{res.content.path}`,
    pagesUrl: `https://${owner}.github.io/${repo}/${res.content.path}`,
    commitSha: res.commit.sha,
  },
};
```

---

6) Boas práticas e dicas

- **Nomes de arquivos**: evite espaços e acentos; use `meu-arquivo.pdf`.
- **Segurança**: não hardcode o token no fluxo. No n8n, crie **Credentials > HTTP Header Auth** com `Authorization: Bearer <TOKEN>` e selecione no **HTTP Request**.
- **Limites**: repositórios públicos aceitam arquivos grandes, mas evite tamanhos muito altos para não estourar limites de banda do Pages.
- **Cache**: GitHub Pages faz cache. Se sobrescrever o mesmo nome, pode demorar um pouco para refletir; force refresh com `Ctrl+F5`.

---

7) Erros comuns (e como resolver)

- **401 Unauthorized**: token inválido/expirado ou header `Authorization` ausente.
- **403 Forbidden**: token sem permissão de **Contents: Read/Write** ou repo errado.
- **404 Not Found**: `owner` / `repo` / `branch` incorretos ou o token não enxerga o repo.
- **422 Unprocessable Entity**: já existe um arquivo nesse caminho; inclua `sha` para sobrescrever ou mude o nome.

Referências (GitHub Docs)

- REST API – **Repository contents** (Create/Update file): docs oficiais.

• **GitHub Pages** – Quickstart e configuração do branch/folder.
• **Tokens** – gerenciamento e permissões para fine-grained tokens.
• Formato de **Raw URL** (exemplos).