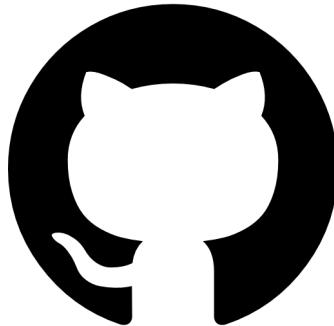


1. Introdução ao GitHub

- O que é GitHub?

GitHub é uma plataforma de hospedagem de código-fonte e controle de versão que usa o Git. Ele permite que desenvolvedores de software colaborem em projetos de forma eficiente e organizada. GitHub oferece uma interface web, além de funcionalidades avançadas como rastreamento de problemas, integração contínua, e muito mais.



- História e evolução do GitHub

GitHub foi fundado em 2008 por Tom Preston-Werner, Chris Wanstrath, PJ Hyett e Scott Chacon. Desde então, a plataforma cresceu exponencialmente e se tornou um dos principais repositórios de código do mundo, hospedando milhões de projetos. Em 2018, a Microsoft adquiriu o GitHub, mas a plataforma continua operando de maneira independente.



- Principais funcionalidades e benefícios

- **Controle de versão:** Usando Git, GitHub permite o rastreamento de alterações no código, facilitando a colaboração e a recuperação de versões anteriores.

- Colaboração: Com funcionalidades como pull requests e revisões de código, GitHub facilita a colaboração entre desenvolvedores.

- Integrações: GitHub se integra com várias ferramentas e serviços, como CI/CD, serviços de nuvem, e muito mais.

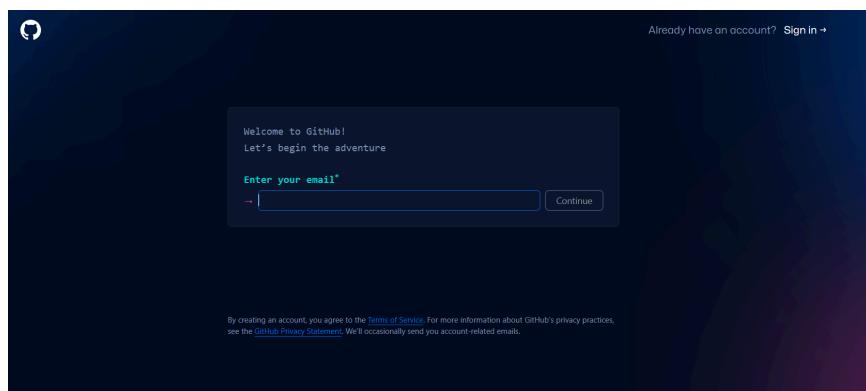
- Documentação: GitHub Pages permite a criação de sites estáticos para documentar projetos diretamente a partir de repositórios.

- Comunidade: GitHub possui uma vasta comunidade de desenvolvedores, facilitando a contribuição para projetos de código aberto.

2. Configuração Inicial

- Criando uma conta no GitHub

1. Acesse o site do GitHub (<https://github.com>) e clique em "Sign up".
2. Preencha o formulário de inscrição com seu email, nome de usuário e senha.
3. Siga as instruções para verificar seu email e configurar seu perfil.



- Instalando o Git e configurando no GitHub

1. Instalando o Git:

- Para Windows: Baixe o instalador do Git em <https://git-scm.com/download/win> e siga as instruções.

- Para macOS: Use Homebrew (`brew install git`) ou baixe diretamente de <https://git-scm.com/download/mac>.

- Para Linux: Use o gerenciador de pacotes da sua distribuição (`sudo apt-get install git` para Debian/Ubuntu).

2. Configurando o Git:

- Abra o terminal e configure seu nome de usuário e email:

```
git config --global user.name "Seu Nome"  
git config --global user.email "seu.email@example.com"
```

```
$ git config --global user.name "Fulano de Tal"  
$ git config --global user.email fulanodetal@example.br
```

A primeira linha de código serve para configurar o nome e a segunda para configurar o email.

- Primeiros passos: criando e clonando repositórios

1. Criando um repositório:

- No GitHub, clique em "New repository".
- Preencha o nome do repositório e uma descrição opcional.
- Escolha se o repositório será público ou privado e clique em "Create repository".

The screenshot shows the 'Create a new repository' interface on GitHub. It includes fields for 'Owner' (set to FlavioOliveiraNeto), 'Repository name' (empty), and 'Description (optional)' (empty). There are two radio button options for visibility: 'Public' (selected) and 'Private'. Under 'Initialize this repository with:', there is a checkbox for 'Add a README file' (unchecked) and a note about writing a long description. A section for '.gitignore' templates shows 'None' selected. A note about ignoring files is present. A 'Choose a license' dropdown is set to 'None'. A note about licenses is also present. At the bottom right is a large green 'Create repository' button.

2. Clonando um repositório:

- No terminal, use o comando `git clone` seguido da URL do repositório:

```
git clone https://github.com/usuario/repositorio.git
```

```
$ git clone git://git.kernel.org/pub/scm/.../linux.git my-linux
$ cd my-linux
$ make
```

3. Comandos Básicos do Git

- Estrutura de um repositório Git

Um repositório Git contém todas as informações necessárias para o histórico do projeto, incluindo:

- **Working Directory:** Diretório onde você trabalha com seus arquivos.
- **Staging Area:** Área onde você prepara suas mudanças antes de fazer commit.
- **Repository:** Onde os commits são armazenados.

- Iniciando um repositório

Para iniciar um novo repositório Git:

```
git init
```

- Principais comandos

1. **git init:** Inicializa um novo repositório Git.
2. **git add:** Adiciona mudanças à staging area.
3. **git commit:** Faz um commit das mudanças na staging area.

```
git commit -m "Mensagem do commit"
```

4. **git push:** Envia os commits para o repositório remoto.

```
git push origin main
```

5. **git pull:** Atualiza o repositório local com mudanças do repositório remoto.

```
git pull origin main
```

- Gerenciamento de branches

1. Criando uma branch:

```
git checkout -b nova-branch
```

2. Mudando para outra branch:

```
git checkout main
```

3. Listando branches:

```
git branch
```

4. Excluindo uma branch:

```
git branch -d branch-antiga
```

4. Trabalho Colaborativo

- Clonando e forkeando repositórios

- **Clonar:** `git clone URL_DO_REPOSITORIO`

- **Fork:** Na página do repositório no GitHub, clique em "Fork".

- Pull requests: como criar e gerenciar

1. Criar um pull request:

- Faça suas alterações em uma nova branch.

- Envie a branch para o GitHub: `git push origin nova-branch`.

- No GitHub, vá até o repositório e clique em "New pull request".

2. Gerenciar pull requests:

- Revise as mudanças propostas.

- Deixe comentários e solicite alterações, se necessário.

- Faça o merge do pull request após aprovação.

- Revisão de código e merge de pull requests

- **Revisão de código:** Deixe comentários e sugestões nas linhas de código alteradas.

- **Merge de pull request:** Após a aprovação, clique em "Merge pull request" no GitHub.

- Resolvendo conflitos

1. Identifique os arquivos em conflito.
2. Edite os arquivos para resolver os conflitos.
3. Adicione os arquivos resolvidos à staging area:

```
git add ARQUIVO_RESOLVIDO
```

4. Faça um commit das mudanças:

```
git commit -m "Resolvido conflito no arquivo"
```

5. Funcionalidades Avançadas

- GitHub Actions: automatizando fluxos de trabalho

GitHub Actions permite automatizar fluxos de trabalho de desenvolvimento, como CI/CD. Crie um arquivo ` `.github/workflows/ci.yml` no seu repositório com a configuração do workflow.

- Issues e Projects: gerenciamento de tarefas e projetos

- **Issues:** Utilize para rastrear bugs, tarefas e solicitações de features.
- **Projects:** Use para organizar tarefas em quadros Kanban.

- GitHub Pages: criando sites estáticos com GitHub

1. Criar um repositório para o site:

- Nomeie o repositório como ` `usuario.github.io` .

2. Adicionar arquivos do site e fazer o push:

- Adicione os arquivos HTML/CSS/Javascript e envie para o repositório.

```
git add .
git commit -m "Adicionando site"
git push origin main
```

- Integrações e APIs

GitHub oferece várias integrações com serviços externos e uma API robusta para automatizar tarefas e acessar dados do repositório programaticamente.

6. Boas Práticas e Dicas

- Escrevendo bons commits e mensagens

- **Commits pequenos:** Faça commits frequentemente e mantenha-os pequenos.

- Mensagens descritivas: Escreva mensagens de commit que descrevam claramente as mudanças.

- Estrutura organizacional de repositórios
 - Organize arquivos por funcionalidade ou módulo.
 - Use pastas para separar código-fonte, testes, documentação, etc.
- Segurança e permissões
 - Proteja as branches principais (main/master).
 - Use equipes e permissões para controlar acesso.
- Uso de templates e arquivos de configuração
 - .gitignore: Liste arquivos e pastas para o Git ignorar.
 - README.md: Documente o propósito e como usar o projeto.
 - CONTRIBUTING.md: Diretrizes para contribuição ao projeto.

- Exercícios para fixação:

Serão feitos 1 exercício para cada tópico mencionado acima.

● **Exercício 1: Configuração Inicial**

Configurar uma conta no GitHub, instalar o Git, criar um repositório local e povoar esse repositório com um projeto de sua autoria ou clonar algum projeto para esse repositório.

● **Exercício 2: Comandos Básicos do Git**

A partir desse repositório criado, fazer alterações no código a fim de praticar os principais comandos do Git.

● **Exercício 3: Trabalho Colaborativo**

Novamente neste repositório criado, criar uma branch, fazer alterações e criar um pull request.

● **Exercício 4: GitHub Pages**

Criar um site estático usando GitHub Pages.

● **Exercício 5: Boas Práticas e Dicas**

Criar a documentação do projeto e adicionar a ele os seguintes arquivos: README.md, .gitignore e CONTRIBUTING.md.