Manual do GitHub

Introdução ao GitHub

O que é GitHub?

GitHub é uma plataforma de hospedagem de código-fonte e controle de versão usando Git. Ele permite que desenvolvedores e equipes colaborem em projetos de software de forma eficiente. O GitHub fornece uma interface web e funcionalidades adicionais, como gestão de projetos, integração contínua e ferramentas de revisão de código.

Git vs. GitHub:
Git: Sistema de controle de versão distribuído que rastreia mudanças em arquivos.
GitHub: Serviço de hospedagem para repositórios Git, oferecendo funcionalidades de colaboração e gerenciamento de projetos.

História e evolução do GitHub

Origem: Criado em 2008 por Tom Preston-Werner, Chris Wanstrath, PJ Hyett e Scott Chacon.
Marcos Históricos:
- 2012: Lançamento do GitHub Enterprise.
- 2018: Aquisição pelo Microsoft.
- Desenvolvimento contínuo de novas funcionalidades como GitHub Actions e GitHub Codespaces.

Principais funcionalidades e benefícios

Controle de Versão: Rastreia alterações e permite reverter para versões anteriores.
Colaboração: Facilitado por pull requests, revisões de código e issues.
Automação: Através de GitHub Actions.
Integração: Compatível com diversas ferramentas e serviços externos.

Configuração Inicial

Criando uma conta no GitHub

1. Acesse github.com.
2. Clique em "Sign up".
3. Preencha os dados requisitados (username, email, password).
4. Complete o processo de verificação.
5. Configure seu perfil inicial.

Instalando o Git e configurando no GitHub

Instalação do Git:
- Windows: Baixe de git-scm.com e siga o instalador.
- macOS: Use Homebrew (brew install git).
- Linux: Use o gerenciador de pacotes (sudo apt install git).

Configuração inicial:
git config --global user.name "Seu Nome"
git config --global user.email "seu.email@exemplo.com"

Primeiros passos: criando e clonando repositórios

Criando um novo repositório:
- Na página inicial do GitHub, clique em "New repository".
- Nomeie o repositório e escolha se será público ou privado.

Clonando repositórios existentes:
git clone https://github.com/usuario/repositorio.git

Comandos Básicos do Git

Estrutura de um repositório Git

Diretórios e arquivos importantes:
- .git: Diretório que contém todos os arquivos de controle de versão.
- README.md: Documento de introdução do projeto.
- .gitignore: Lista de arquivos/diretórios a serem ignorados pelo Git.

Iniciando um repositório

Comando git init:
git init

Principais comandos

Adicionar arquivos (git add):
git add arquivo.txt

Commitar mudanças (git commit):
git commit -m "Mensagem de commit"

Enviar mudanças para o repositório remoto (git push):
git push origin main

Puxar mudanças do repositório remoto (git pull):
git pull origin main

Gerenciamento de branches

Criando uma nova branch:
git branch nome-da-branch

Alternando entre branches:
git checkout nome-da-branch

Merge de branches:
git merge nome-da-branch

Trabalho Colaborativo

Clonando e forkeando repositórios

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

Forkeando:
- Na página do repositório, clique em "Fork".

Pull requests: como criar e gerenciar

Criando um pull request:
- Na página do repositório, vá para a aba "Pull requests" e clique em "New pull request".
- Compare as branches e clique em "Create pull request".

Gerenciando pull requests:
- Comente e revise o código proposto.
- Use a interface para aprovar ou solicitar mudanças.

Revisão de código e merge de pull requests

Revisão de código:
- Analise as alterações e comente se necessário.

Realizando o merge:
- Após a aprovação, clique em "Merge pull request".

Resolvendo conflitos

Identificação de conflitos:
- O Git avisa sobre conflitos durante o merge.

Resolução:
- Edite os arquivos conflitantes e marque as alterações resolvidas.
git add arquivo.txt
git commit

Funcionalidades Avançadas

GitHub Actions: automatizando fluxos de trabalho

Configuração de workflows:
- Crie um arquivo .github/workflows/main.yml.

Exemplo de workflow:

```
name: CI
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Run a one-line script
      run: echo Hello, world!
```

Issues e Projects: gerenciamento de tarefas e projetos

Criando e gerenciando issues:
- Na aba "Issues", clique em "New issue".

Usando projetos:
- Na aba "Projects", crie um novo projeto e organize issues e tarefas.

GitHub Pages: criando sites estáticos com GitHub

Configuração:
- No repositório, vá para "Settings" > "Pages".
- Escolha a branch e o diretório para a publicação.

Exemplo de site com Jekyll:
- Use templates Jekyll para criar páginas estáticas.

Integrações e APIs

Integrações:
- Conecte GitHub a outras ferramentas como Slack e Trello.

Utilizando a API do GitHub:
- Acesse api.github.com para detalhes da API.
- Exemplos de uso com cURL:

```
curl https://api.github.com/users/usuario
```

Boas Práticas e Dicas

Escrevendo bons commits e mensagens

Boas práticas:
- Mensagens claras e concisas.
- Utilize o formato:
Resumo curto (50 caracteres ou menos)

Descrição detalhada, se necessário.

Estrutura organizacional de repositórios

Organização de arquivos e pastas:
- Mantenha uma estrutura lógica e limpa.
- Utilize arquivos como README.md para documentação.

Segurança e permissões

Configuração de permissões:
- Gerencie acessos na aba "Settings" > "Collaborators & teams".

Boas práticas de segurança:
- Utilize autenticação de dois fatores (2FA).
- Evite expor chaves e senhas no código.

Uso de templates e arquivos de configuração

.gitignore:
- Liste arquivos/diretórios a serem ignorados pelo Git.

Templates de issues e pull requests:
- Crie arquivos .github/ISSUE_TEMPLATE.md e .github/PULL_REQUEST_TEMPLATE.md para padronizar.