CONHECENDO GITHUB

1. Repositórios (Repositories)

Um repositório no GitHub é o local onde o código do seu projeto é armazenado. Ele contém todos os arquivos do projeto, incluindo o histórico de versões e alterações feitas ao longo do tempo. Os repositórios podem ser públicos (visíveis para qualquer pessoa) ou privados (acessíveis apenas para pessoas autorizadas). Para colaborar com outras pessoas ou compartilhar seu projeto com o mundo, um repositório público é a opção ideal.

Dicas:

- Estruture o repositório organizando arquivos em diretórios.
- Use arquivos auxiliares como `README.md` e `LICENSE` para tornar o repositório informativo e transparente.

2. Commits

Commits são o coração do controle de versão. Cada vez que você faz alterações no projeto, pode "comitar" essas mudanças, salvando-as com uma mensagem que descreve o que foi alterado. Isso cria um ponto no histórico do repositório ao qual você pode voltar se precisar.

Boas práticas:

- Faça commits frequentemente, com pequenas mudanças incrementais.
- Use mensagens de commit claras e descritivas para facilitar o rastreamento de alterações (exemplo: "Corrigido bug no formulário de login").

3. Branches

Branches permitem trabalhar em várias funcionalidades ou correções simultaneamente sem interromper o desenvolvimento principal. A branch principal é normalmente chamada de `main` ou `master`, e outras branches são criadas para diferentes tarefas (como adicionar um novo recurso ou corrigir um bug).

Como usar:

- Crie uma nova branch para cada nova funcionalidade: `git checkout -b nome-da-feature`.
- Depois que o trabalho estiver completo, faça um merge da branch com a `main`.

Vantagens:

- Evita conflitos entre diferentes funcionalidades.
- Facilita o controle de diferentes versões do projeto.

4. Pull Requests (PRs)

Pull Requests são usados para integrar as alterações feitas em uma branch ao código principal do projeto. No GitHub, PRs são uma maneira de colaborar em equipe, pois permitem a revisão de código e a discussão sobre as mudanças antes de serem incorporadas.

Processo típico:

- Um desenvolvedor faz alterações em uma branch e cria um PR.
- Outros membros da equipe revisam o código e fazem sugestões ou aprovações.
- Após a aprovação, o PR é mesclado com a branch principal.

Dicas:

- Revise e teste o código antes de abrir um PR.
- Utilize a seção de comentários para discutir o impacto das mudanças.

5. Issues

As issues funcionam como um sistema de gerenciamento de tarefas e bugs no GitHub. Elas permitem que você rastreie problemas, solicite novas funcionalidades ou faça anotações sobre o progresso do projeto.

Melhores práticas:

- Use títulos e descrições claras para suas issues.
- Categorize as issues com etiquetas (labels) como "bug", "feature", ou "enhancement".
- Atribua as issues a membros da equipe para facilitar o gerenciamento.

6. Forks

Um fork é uma cópia completa de um repositório que permite que você faça alterações sem interferir no repositório original. Isso é útil quando você deseja contribuir com um projeto de código aberto ou criar uma versão personalizada de um repositório.

Exemplo:

- Ao contribuir para projetos open-source, você primeiro faz um fork do projeto original, faz as mudanças no seu fork e, depois, abre um PR para sugerir suas alterações ao repositório original.

7. README.md

O arquivo `README.md` é o cartão de visita do repositório. Ele fornece uma descrição do projeto, como ele funciona, como configurar o ambiente e qualquer outra informação que ajude novos usuários ou colaboradores a entenderem o propósito do projeto.

Estrutura básica do README:

- Título do projeto.
- Descrição clara do objetivo.
- Instruções de instalação.
- Exemplos de uso.
- Contribuições e licenciamento.

O arquivo usa a sintaxe **Markdown**, o que permite adicionar formatação simples, como cabeçalhos, listas e links.

8. Clonar e Git Local

Ao clonar um repositório, você cria uma cópia local do projeto para trabalhar diretamente no seu computador. Isso permite desenvolver e testar o código antes de enviá-lo de volta para o GitHub.

Comandos básicos:

- `git clone URL-do-repositório`: clona o repositório para sua máquina local.
- `git add .`: adiciona as mudanças ao "staging".
- `git commit -m "Mensagem do commit"`: grava as mudanças no repositório local.
- `git push`: envia as alterações para o repositório no GitHub.

9. GitHub Actions

GitHub Actions é uma poderosa ferramenta de automação. Com ela, você pode configurar pipelines que automatizam tarefas, como execução de testes automatizados, deploys, ou até verificação de estilo de código. As actions são configuradas através de arquivos YAML, onde você define os passos que devem ser executados.

Exemplo:

- Criar uma Action que execute testes sempre que um PR for aberto.

10. GitHub Pages

GitHub Pages é um serviço que permite hospedar sites estáticos diretamente de um repositório. É frequentemente usado para sites de portfólio, documentação ou páginas de projetos.

Como usar:

- Configure o site na aba "Settings" do repositório.
- Coloque arquivos HTML, CSS e JavaScript na branch configurada (geralmente `gh-pages` ou `main`).
- O GitHub criará uma URL onde o site estará disponível.