Git and github

O que é Git?

Git é um sistema de controle de versão distribuído usado para gerenciar projetos de software. Ele permite que várias pessoas trabalhem no mesmo projeto ao mesmo tempo sem sobrescrever o trabalho uns dos outros. Com o Git, você pode rastrear alterações em seu código ao longo do tempo, restaurar versões anteriores do código e colaborar facilmente com outros desenvolvedores.

Como instalar o Git?

Você pode instalar o Git em seu computador seguindo as instruções fornecidas no site oficial: https://git-scm.com/downloads

Comandos Básicos

Aqui estão alguns dos comandos básicos mais comuns no Git:

1. `git init` - Inicializa um repositório Git vazio em seu diretório de trabalho atual.
2. `git clone` - Copia um repositório Git existente em sua máquina local.
3. `git add` - Adiciona arquivos ao índice Git (staging area) para prepará-los para o commit.
4. `git commit` - Salva as mudanças feitas nos arquivos adicionados no índice Git (staging area).
5. `git push` - Envia as alterações locais para um repositório remoto no GitHub, GitLab ou outro.
6. `git pull` - Atualiza seu repositório local com as alterações feitas em um repositório remoto.
7. `git status` - Exibe o estado atual do repositório, incluindo arquivos modificados, arquivos adicionados e arquivos prontos para commit.
8. `git log` - Exibe um registro de todos os commits feitos no repositório.
9. `git branch` - Exibe todas as branches (ramificações) em seu repositório local.
10. `git checkout` - Muda para uma branch específica em seu repositório local.
11. `git merge` - Combina duas branches diferentes em uma só.

Exemplos de uso

Inicializando um novo repositório:

1. Abra o terminal e navegue até a pasta do seu projeto.

2. Digite `git init` para criar um novo repositório.

Adicionando arquivos ao índice Git:

1. Digite `git add nome_do_arquivo` para adicionar um arquivo específico.
2. Digite `git add .` para adicionar todos os arquivos no diretório atual.

Fazendo um commit:

1. Digite `git commit -m "mensagem do commit"` para fazer um commit com uma mensagem específica.

Enviando alterações para um repositório remoto:

1. Digite `git push origin nome_da_branch` para enviar as alterações locais para um repositório remoto em uma branch específica.

Clonando um repositório existente:

1. Digite `git clone url_do_repositorio` para clonar um repositório Git existente em sua máquina local.

Atualizando seu repositório local com as alterações feitas em um repositório remoto:

1. Digite `git pull origin nome_da_branch` para atualizar seu repositório local com as alterações feitas em um repositório remoto em uma branch específica.

Exibindo o estado atual do seu repositório:

Para exibir o estado atual do seu repositório Git, você pode usar o comando `git status`. Esse comando exibe informações sobre o estado atual dos arquivos no seu repositório, incluindo:

- Arquivos que foram modificados, mas ainda não foram preparados para commit
- Arquivos que foram preparados para commit, mas ainda não foram commitados
- Arquivos que não estão sendo rastreados pelo Git

A saída do comando `git status` geralmente inclui sugestões sobre as etapas que você precisa tomar para gerenciar esses arquivos no seu repositório. Por exemplo, se houver

arquivos modificados, o Git sugerirá que você os prepare para commit usando o comando `git add`.

Fluxos:

Existem vários tipos de fluxo de trabalho (ou workflow) do Git, mas os mais comuns são:

1. **Git Flow** - Um fluxo de trabalho popular para projetos grandes e complexos. Ele usa duas branches principais (master e develop) e várias branches de recursos (feature branches) para o desenvolvimento de novas funcionalidades.
2. GitHub Flow - Um fluxo de trabalho simples e fácil de usar que usa apenas uma branch principal (geralmente a branch master) e usa as pull requests para gerenciar as mudanças.
3. GitLab Flow - Um fluxo de trabalho similar ao GitHub Flow, mas com uma ênfase maior em pipelines de integração contínua (CI/CD) e automação.
4. GitLab Flow - Um fluxo de trabalho similar ao GitHub Flow, mas com uma ênfase maior em pipelines de integração contínua (CI/CD) e automação.
5. Feature Branch Workflow - Um fluxo de trabalho simples que usa branches de recursos para o desenvolvimento de novas funcionalidades. Cada branch de recurso é criada a partir da branch principal e é excluída assim que a funcionalidade é concluída.

Cada fluxo de trabalho tem suas próprias vantagens e desvantagens e o melhor fluxo a ser usado depende do tamanho e da complexidade do projeto, bem como da equipe de desenvolvimento e das suas necessidades.

Plataformas online:

Existem diversas plataformas que permitem praticar o uso do Git em projetos reais, tanto para trabalhar em colaboração com outros desenvolvedores quanto para desenvolver projetos individuais. Algumas delas são:

1. GitHub - É a plataforma mais conhecida para hospedar repositórios Git, oferecendo ferramentas de gerenciamento de projetos, colaboração em equipe e integração com outras ferramentas de desenvolvimento.
2. GitLab - É uma plataforma similar ao GitHub, que permite hospedar repositórios Git, gerenciar projetos, fazer integração contínua (CI/CD) e colaborar em equipe.
3. Bitbucket - Outra plataforma de hospedagem de repositórios Git, com recursos de gerenciamento de projetos, colaboração em equipe e integração com outras ferramentas.
4. SourceForge - Plataforma de hospedagem de projetos de software livre, que permite hospedar repositórios Git, gerenciar projetos, colaborar em equipe e disponibilizar ferramentas para desenvolvedores.

5. Gitpod - Ambiente de desenvolvimento online que permite desenvolver e testar projetos Git em um ambiente isolado, com recursos de integração contínua e colaboração em equipe.

Aplicativos clientes:

Existem vários aplicativos cliente que permitem trabalhar com o Git em um ambiente gráfico e mais amigável para o usuário. Alguns dos mais populares são:

1. GitHub Desktop - um aplicativo oficial do GitHub que oferece uma interface gráfica fácil de usar para trabalhar com repositórios Git no Windows e no Mac.
2. Sourcetree - um cliente Git gratuito da Atlassian, que oferece uma interface gráfica amigável para gerenciar repositórios Git no Windows e no Mac.
3. GitKraken - um cliente Git com interface gráfica atraente e intuitiva, que permite gerenciar repositórios Git no Windows, Mac e Linux.
4. Git Cola - um cliente Git gratuito e de código aberto com uma interface gráfica simples e fácil de usar, disponível no Windows, Mac e Linux.
5. TortoiseGit - um cliente Git gratuito que se integra ao Windows Explorer e oferece uma interface gráfica para gerenciar repositórios Git.

O fluxo que iremos usar nos projetos: Git Flow

Thiago André Cardoso Silva
twitter/instagram: **@programador_who**