Guia de comandos Git

Sobre o material

Este material aborda os conceitos fundamentais do versionamento de código com o Git listando os comandos básicos e avançados, conceitos sobre a plataforma de hospedagem de código Github e como se inscriver no programa Github Student.

Sumário

O que é Git?

Fonte: Wikipédia

Desenvolvido por Linus Torvalds (criador do kernel do Linux) em 2005 surge um sistema de controle de versões que, futuramente, seria reconhecido e utilizado globalmente.

Tudo começou com o rompimento de relações entre a comunidade que desenvolvia o kernel do Linux e a BitKeeper.Com esse rompimento, a ferramenta do BitKeeper passou a ser paga.

Com tudo isso, Linus Torvalds decidiu construir um sistema de controle de versão que tivesse a melhor performance e usou a experiência que teve com a BitKeeper para construir o Git.

Devido seu grande sucesso no mercado várias empresas começaram a criar plataformas de hospedagem de código-fonte e arquivos utilizando o Git como controle de versão, um exeplo de plataforma seria o Github, que iremos abordar logo abaixo.

Comandos Git Básicos

Os comandos Git básicos são essenciais para começar a usar o Git como ferramenta de controle de versão. Aqui estão alguns comandos comuns com exemplos:

- Clonar um repositório remoto para o seu ambiente local. Exemplo: `git clone https://github.com/seu-usuario/seu-repositorio.git`
- Criar e gerenciar branches para trabalhar em diferentes fluxos de desenvolvimento. Exemplo: `git branch implementacao-login`
- Alternar entre diferentes branches ou versões do código. Exemplo: `git checkout implementacao-login`
- Adicionar arquivos ou alterações específicas ao próximo commit. Exemplo: `git add index.html`
- Atualizar o repositório local com as alterações mais recentes do repositório remoto. Exemplo: `git pull origin implementacao-login`
- Enviar suas alterações locais para o repositório remoto. Exemplo: `git push origin implementacao-login`
- Buscar as alterações mais recentes do repositório remoto sem mesclá-las ao código local. Exemplo: `git fetch origin`

Comandos Git Avançados

Além dos comandos básicos, existem comandos Git mais avançados que podem ser úteis durante o desenvolvimento de software. Aqui estão alguns exemplos:

- Modificar o commit mais recente com novas alterações ou uma nova mensagem de commit. Exemplo: `git commit --amend -m "Mensagem corrigida!"`
- Configurar variáveis de configuração do Git. Exemplo: `git config --global user.name "Seu Nome"`

  Adicionar configurações pra apenas um commit

  - É possível setar configurações as configurações e realizar o commit em uma só linha (pra não ter que setar globalmente e correr o risco de alguém pegar a máquina e commitar no seu nome).
  - Exemplo: `git -c "user.name=Gabriel Silva" -c "user.email=silva@gmail.com" commit -m "Mensagem do commit"`

- Gerenciar repositórios remotos e suas configurações. Exemplo: `git remote add origin <URL-do-repositório-remoto>`, `git remote remove origin`, `git remote`
- Exibir o histórico completo de referências, incluindo commits que não estão mais visíveis. Exemplo: `git reflog`
- Exibir o histórico de commits com detalhes, incluindo autor, data e mensagem. Exemplo: `git log`
- Desfazer um commit específico, criando um novo commit que reverte as alterações. Exemplo: `git revert <hash-do-commit>`

- Armazenar temporariamente alterações não commitadas para trabalhar em outra tarefa. Exemplo: `git stash`, `git stash save "Nome do stash"`

  Comandos relacionados ao git stash:

  - Exibir a lista de stashes disponíveis. Exemplo: `git stash list`
  - Aplicar e remover o stash mais recente da pilha. Exemplo: `git stash pop`
  - Aplicar o stash mais recente sem removê-lo da pilha. Exemplo: `git stash apply`
  - Aplicar um stash específico da pilha usando um índice n. Exemplo: `git stash apply stash@{2}`
  - Remover um stash específico da pilha usando um índice n. Exemplo: `git stash drop stash@{1}`
  - Exibir as alterações contidas em um stash específico da Exemplo: `git stash show stash@{0}`

- Aplicar um commit específico de uma branch para outra. Exemplo: `git cherry-pick <hash-do-commit>`, `git cherr-ypick -n 2475e9a`

  Diferença entre git stash e git cherrypick:

  - O comando git stash é usado para armazenar temporariamente alterações não commitadas, permitindo que você as recupere mais tarde.
  - O comando git cherrypick é usado para aplicar um commit específico de uma branch para outra.

- Combinar alterações de uma branch para outra. Exemplo: `git merge nome-da-branch`
- Exibir as diferenças entre commits, branches ou arquivos específicos. Exemplo: `git diff branch1 branch2`
- Pesquisar por uma determinada string nos arquivos do repositório. Exemplo: `git grep "palavra-chave"`

Programa GitHub Student

O programa "GitHub Student" é uma iniciativa destinada a estudantes, que oferece benefícios especiais para auxiliar no desenvolvimento de projetos e colaboração no GitHub. Alguns dos benefícios incluem:

- Conta GitHub gratuita com recursos aprimorados.
- Acesso a ferramentas e recursos exclusivos para estudantes.
- Créditos em serviços de hospedagem e outros recursos úteis para projetos.
- Participação em programas de aprendizado e eventos da comunidade.
- Acesso ao Github Student Developer Pack.

Para se inscrever no programa GitHub Student, acesse education.github.com/students e siga as instruções fornecidas.

Padrões de nomenclatura

Para auxiliar no entendimento da equipe de desenvolvimento foi criado padrões de nomenclatura de branchs e commits. Para branchs o padrão que recomendamos é o seguinte:

O nome da branch deverá ser divida em duas partes

1. Prefixo: Categoria da branch, as categorias são as seguintes:
    - feature: para funcionalidades novas
    - fix: para correção de um bug
    - refactor: para refatoração do código
    - doc: para mudanças na documentação
    - style: para mudanças visuais (alteração na fonte, tamanho, cor, etc)

2. Sufixo: Funcionalidade atribuida à branch

Exemplos: feature/cadastro-usuario, style/cor-botao-cadastro

Para nomenclatura de commits uma boa descrição do que foi feito naquele commit já é o suficiente para auxiliar os outros desenvolvedores a entender, de forma prévia, o que foi desenvolvido.

Recursos e referências

- Site oficial do Git
- Pro Git (Livro gratuito sobre Git)
- Wikipédia Git
- Blog sobre Git
- Padrões de nomenclatura

Voltar ao início