



x x x x

GIT FLOW

Rebase, Merge,
Squash, Revert e
Reset Commit

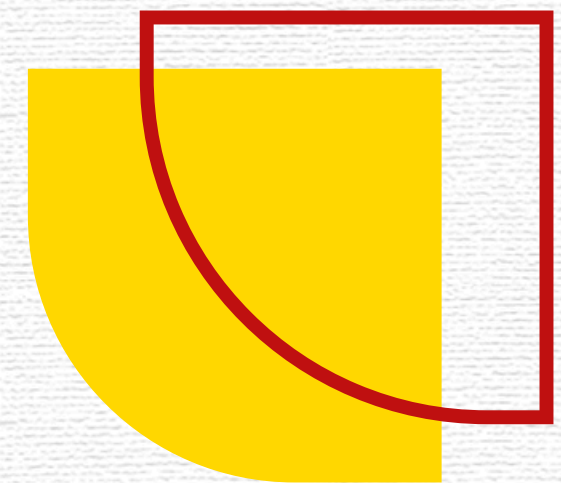
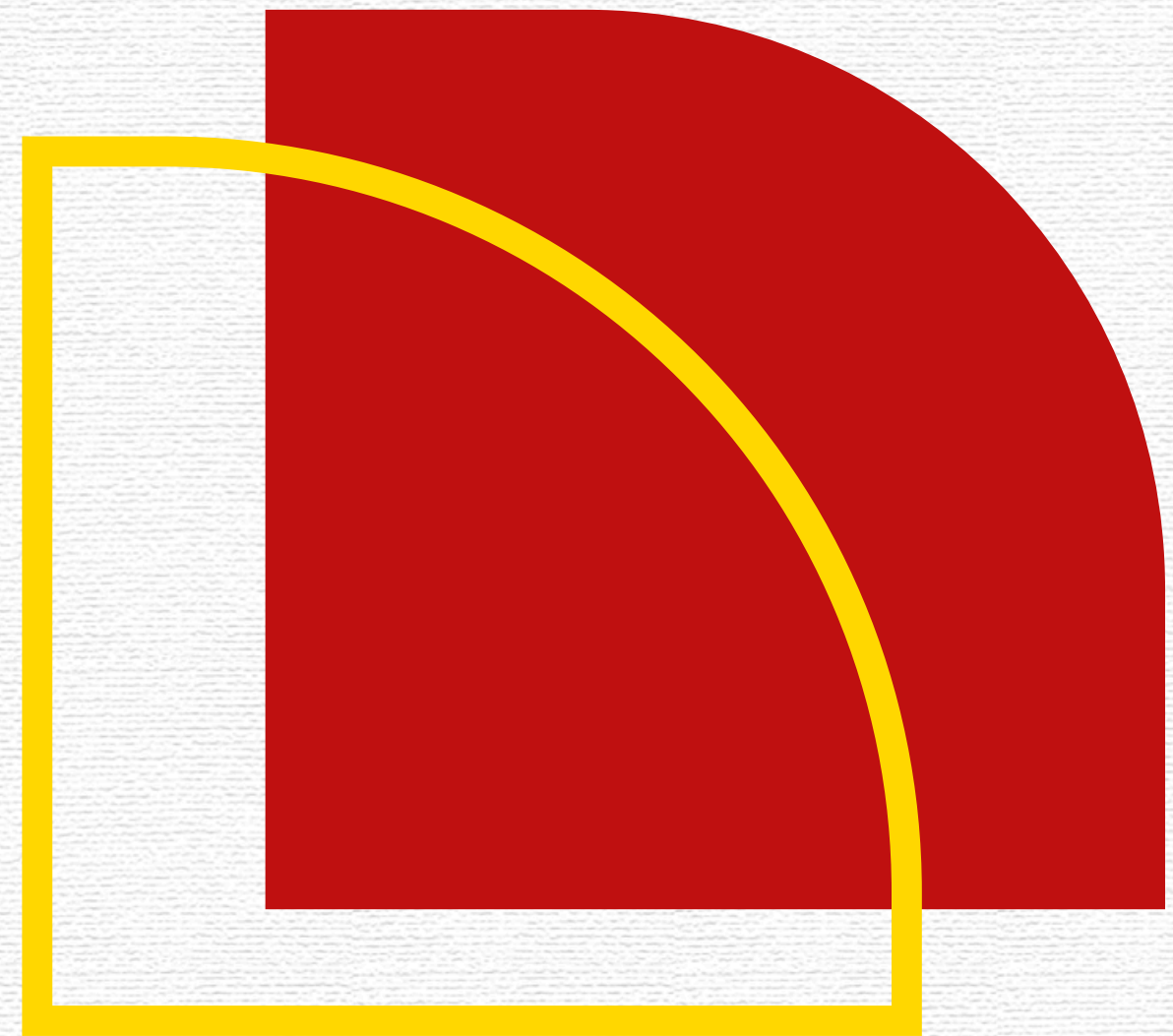
Melhorando o fluxo de commits e merges com Git

x x x x



PROBLEMAS

- Histórico poluído (vários commits pequenos e sem contexto claro)
- Dificuldade para entender o que foi feito em uma feature
- Conflitos frequentes ao fazer merge com a master
- Falta de padrão entre os devs



OBJETIVO

- Manter o histórico limpo e organizado
- Melhorar o entendimento do que foi feito em cada feature
- Evitar conflitos desnecessários
- Padronizar o fluxo do time



FLUXO ATUAL

feature → commit (vários)
→ merge da master (merge commit)
→ histórico bagunçado





O QUE É O GIT MERGE?

- Junta o histórico da branch base com a branch alvo
- Cria um commit de merge
- Preserva o histórico completo e linear
- Fácil de usar, mas pode deixar o log poluído





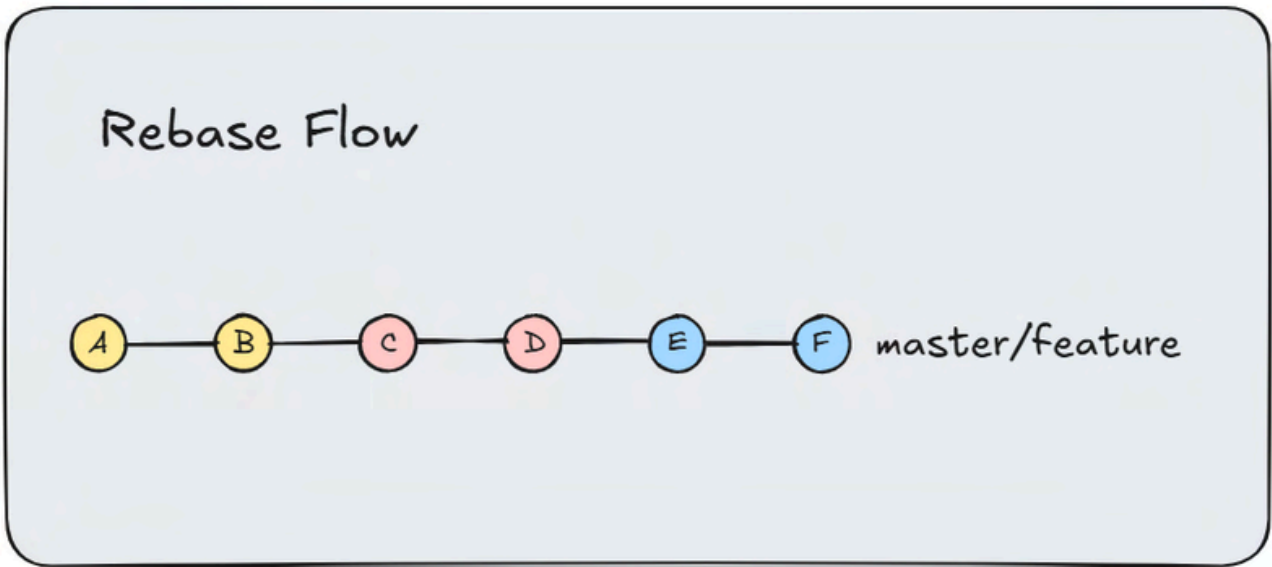
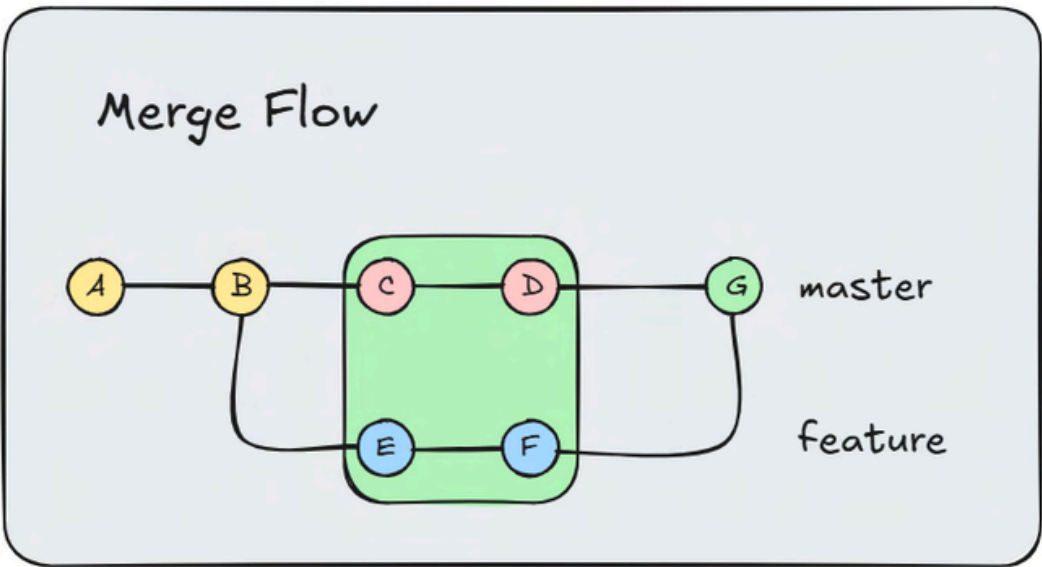
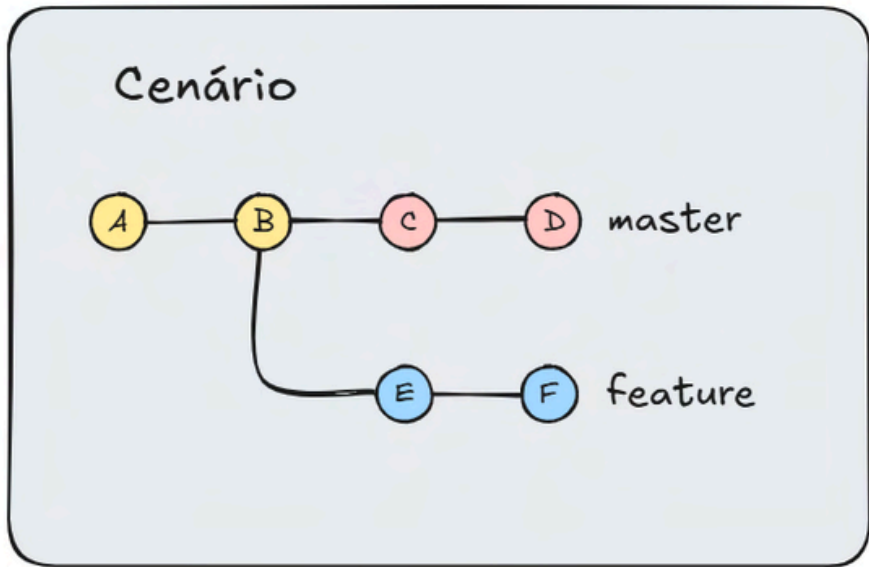
O QUE É O GIT REBASE?

- Reposiciona os commits da sua branch na frente dos commits da base (ex: master)
- Reescreve o histórico
- Garante linearidade no log
- Ideal para branches locais ou não compartilhadas





COMPARAÇÃO VISUAL: MERGE VS REBASE





O QUE É O SQUASH?

- Junta vários commits em um só
- Ideal para “limpar” commits pequenos (fix, ajuste, wip)
- Pode ser usado no rebase interativo (`git rebase -i`) ou na hora de fazer merge no PR





O QUE É O REVERT?

- Desfaz mudanças preservando o histórico
- Vantagem: Mudanças com segurança
- Desvantagem: Gera commits adicionais e possíveis conflitos





O QUE É O RESET?

- Move o ponteiro HEAD para um commit específico
- Tipos de Reset:
 - Soft: mantém mudanças no índice e no diretório de trabalho.
 - Mixed: mantém mudanças no diretório de trabalho, mas limpa o índice
 - Hard: descarta todas as mudanças no índice e no diretório de trabalho.





PADRÃO SUGERIDO

1. Trabalhe normalmente em sua branch
2. Antes do PR:
 - a. `git fetch origin`
 - b. `git rebase origin/master`
 - c. Resolva conflitos (se houver)
 - d. Squash commits pequenos
3. Envie o PR com um commit limpo
4. Após aprovação, merge com fast-forward



Benefícios

- Histórico limpo e linear
- Estrutura da árvore linear
- Fácil de entender o que cada feature entregou
- Menos conflitos
- Melhor rastreabilidade
- Time padronizado





Quando não usar rebase/squash

- Em branches compartilhadas (pode quebrar o trabalho dos outros)
- Quando for necessário preservar o histórico original detalhado





REFERÊNCIAS

- Pro Git Book
- GitHub Docs
- Atlassian Git Tutorials
- Git e Github para iniciantes



DÚVIDAS?





*Esteja aberto a aprender.
Você não sabe tudo e não
está sempre certo.*

