



Commit com
Qualidade



Hello!

Ketlin Pedron

**Desenvolvedora de Software
com pé na Qualidade**



O que é um Commit

Commit refere-se a ideia de fazer permanentes um conjunto de mudanças.

Um commit é o ato de enviar as modificações necessárias para uma base de dados.



Por que é importante?

Em desenvolvimento de software, os commits são essenciais para controle de versão de release.

Em situações de problema, permite fácil reversão de versão para determinado ponto na linha do tempo.



Times de Desenvolvimento



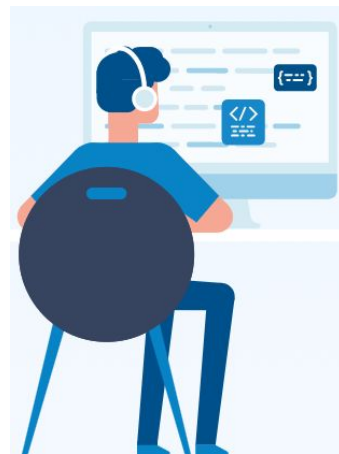
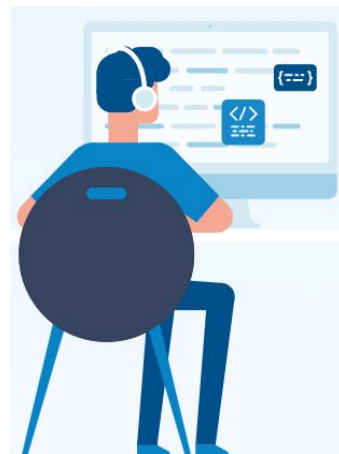
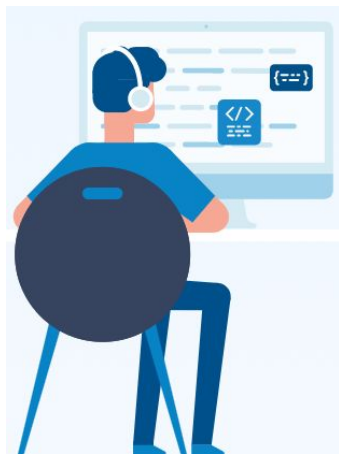


09:00

11:00

12:00

17:00





*GIT é muito mais do que
pull e push*



GIT REBASE

GIT STATUS

GIT MERGE

GIT CONFIG

GIT STASH

GITK BRANCH

git rebase -i HEAD~3
(salvador de vidas)

GIT LOG

GITK -ALL

GIT REMOTE -V

GIT CHECKOUT
(para recuperar o arquivo)





*Precisamos primeiro
entender sobre política de
branch*

O que é Branch?





Existem diferentes formas de organizar repositórios, cada empresa escolhe a sua.

Porém, o **Git Flow** é um bom exemplo de organização e é utilizado por muitos times!





GIT Flow



Branch MASTER: Possui o código mais maduro do projeto, sem bugs. Código em nível de produção.

Branch DEVELOP: Após as features serem finalizadas, serão encaminhadas para a branch develop e testadas. As atualizações que estão prontas no develop serão enviadas para o Master.

Branch FEATURE: Recursos novos para a aplicação serão desenvolvidos em branches feature. Utilizam uma convenção para o nome: feature/new-layout. Este branch deve ser criado a partir do branch DEVELOP. Após finalizado deve ser enviado para o mesmo.



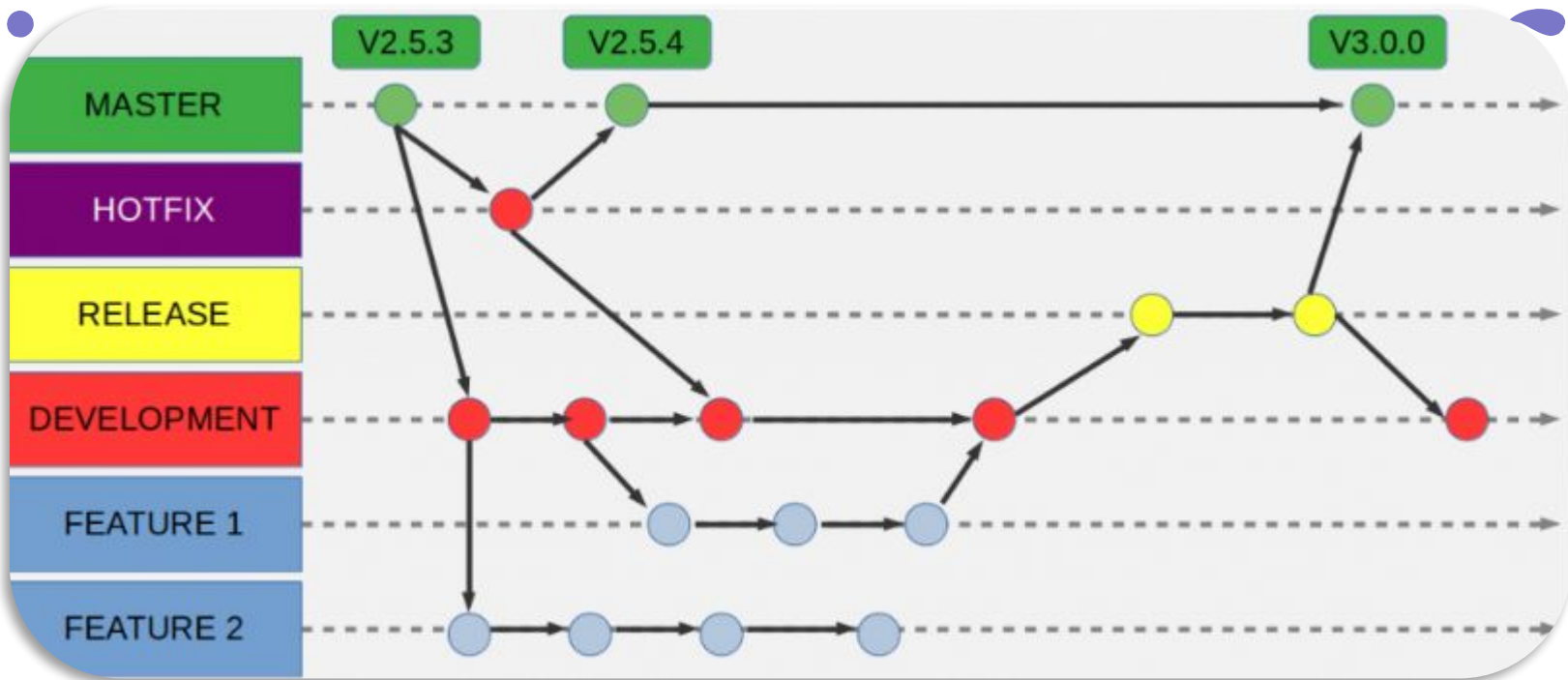


Branch HOTFIX: Onde serão realizadas correções de bugs críticos encontrados na produção. É criado a partir do Master e após a correção do bug a correção é enviada para o DEVELOP e MASTER.

Branch RELEASE: É uma branch intermediária entre o DEVELOP e MASTER. No geral são usados para preparação do lançamento da próxima versão de produção. São permitidas

O git flow além de ser um framework, é uma extensão que pode ser instalada no seu repositório ☺



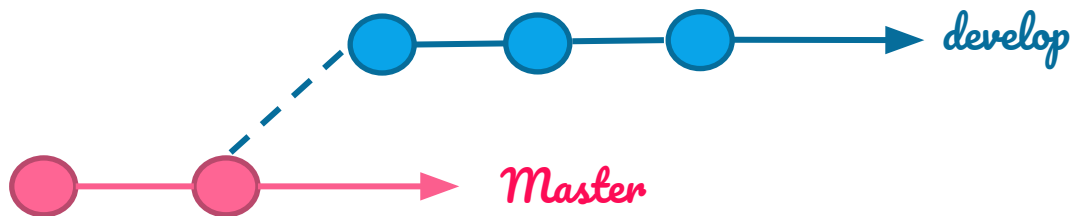
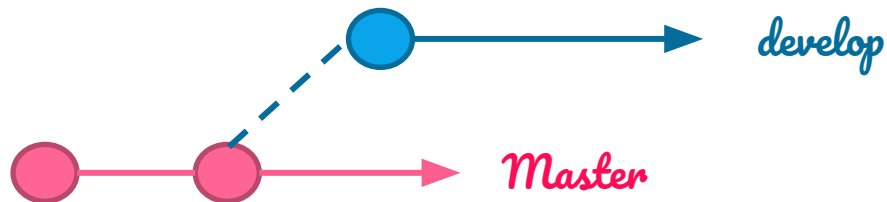




Para fazer alterações entre
branches é essencial saber
utilizar merge e rebase

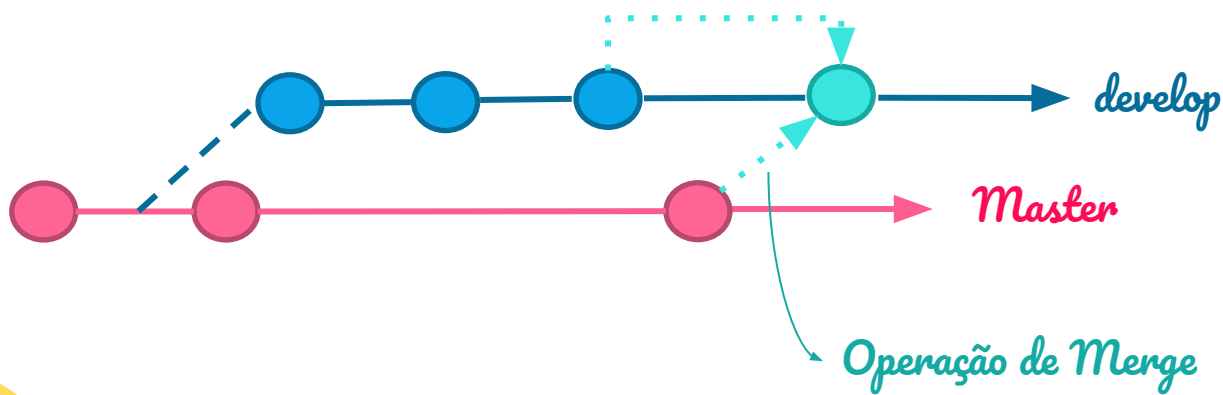
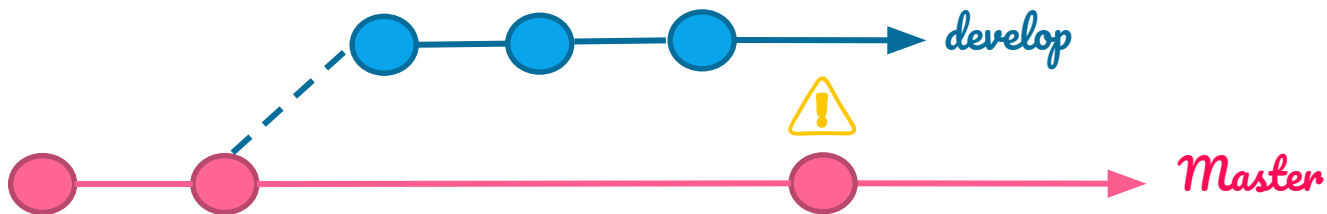


GIT MERGE





GIT MERGE





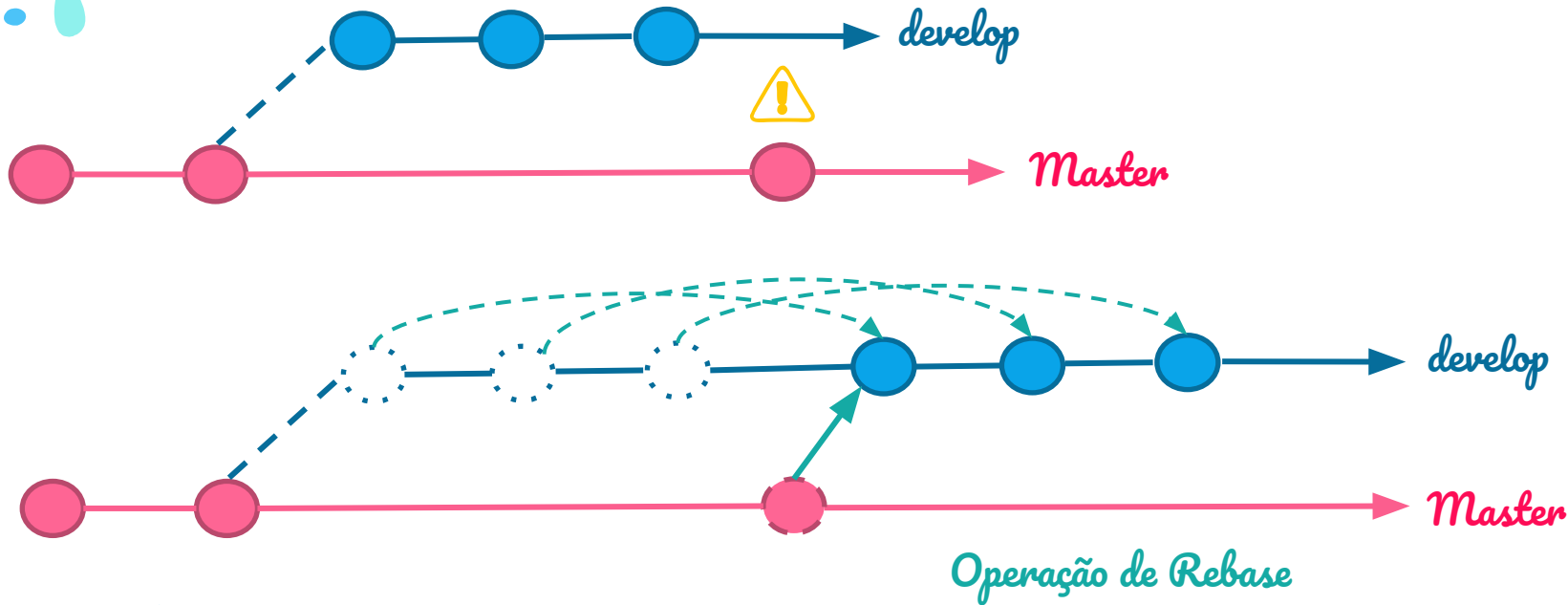
GIT MERGE



Ao realizar um merge, um commit genérico é criado no branch de destino. Esse commit adicional pode prejudicar e dificultar a leitura do histórico dos commits, pois ele agrupa os commits e gera apenas uma mensagem final de commit.



GIT REBASE





GIT Rebase

Ao realizar um rebase, o histórico linear dos commits é mantido, fazendo com que as mensagens originais permaneçam.

Porém, se a linha em desenvolvimento estiver muito desatualizada em relação ao master, caso o rebase não seja executado de forma correta podem ocorrer vários problemas (conflitos) no histórico dos branches envolvidos e inclusive pode acontecer a perda de trabalho durante o processo de mesclagem.

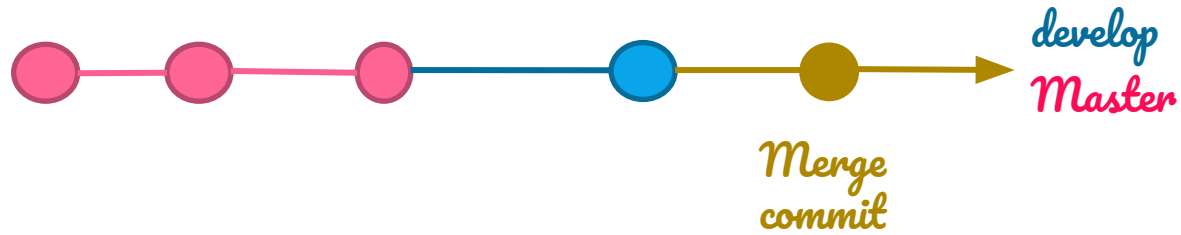
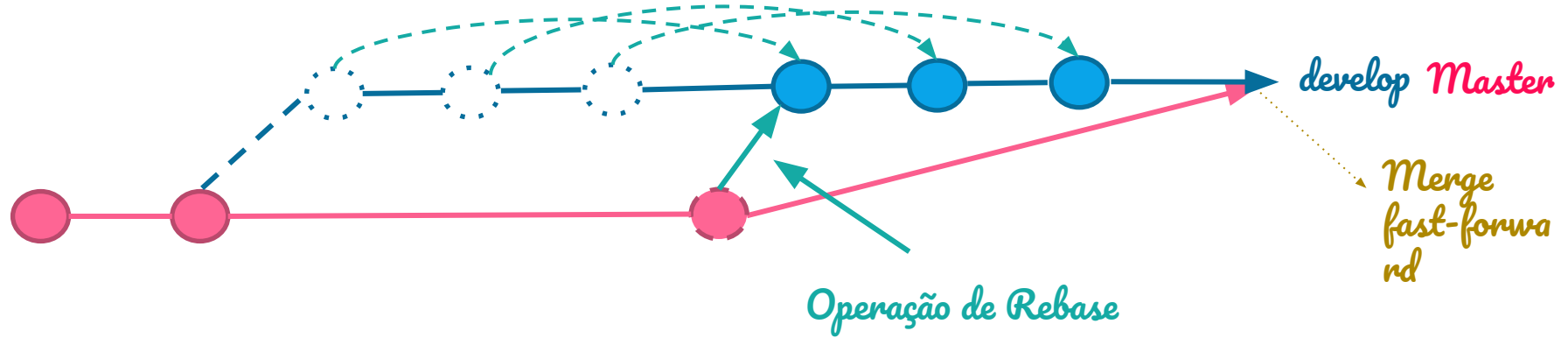


Usar git Merge ou git Rebase?

- Entre branches públicos não é recomendado utilizar o rebase, pois não facilita o “rastreamento” de quando dois branches foram fundidos.
- O rebase é recomendado quando é necessário trazer as últimas modificações do branch principal para o branch develop.
- Quando for necessário enviar commits do branch de develop para o principal, o recomendado é utilizar o merge, para manter rastreáveis as modificações do projeto.



O ideal é utilizar os dois





O que não fazer



 Esperar acabar o dia de trabalho para enviar um commit
Ou o pior: Somente commitar quando finalizar uma feature.



Criar commits que não favorecem a reversão de release



Não configurar adequadamente o arquivo `.git ignore`





*E se duas pessoas editarem o
mesmo Código fonte?*



PREGUIÇA

PRESSA

“Minha equipe é pequena, não é necessário”

“Ninguém vai ler”

“Tem que ser em inglês?”

“Commit direto no Master não dá nada”

“Só um commit dá”



O que irá te tentar?





Boas Práticas





Utilize um padrão nas mensagens:

- Para o título utilize 50 caracteres (ou menos)
- Para facilitar buscas, utilize tags no título
- Para o corpo do texto, utilize 72 caracteres por linha
- Utilize verbos no “imperative”
- Se precisar de mais de um parágrafo, deixe 1 linha em branco
- Utilize o “Signed-off-by” ao final da mensagem



Mantenha o develop atualizado e estável



Só envie para o Master o que tiver sido validado e 100% funcional



Combine rebase e merge



Referencie outros commits se for necessário, para isso use o hash do commit a ser referenciado




Configure seu nome/e-mail para ser exibido no histórico



Peça ajuda sem medo!





Ao criar um commit devemos
sempre pensar em nós
mesmos navegando pelo
histórico.

Vou entender o que fiz?





Thanks!

Perguntas?

@k_pedron

pedron.ketlin@gmail.com

