Comandos para o git - no terminal git Bash ou power shell

Puxar e clonar repositório do GitHub (via comando)

1) vá ao GitHub (remoto) e copia a url do repositório que deseja clonar;
2) no terminal escolha a pasta onde será feita a clonagem;
3) git clone + a url que vc copiou do gitHub (remoto)
4) ls para confirmar o nome do repositório
5) git status para ver a situação dos arquivos

****

Comandos no terminal git bash para atualizar o repositório remoto com o repositório local de sua máquina.

1) vc deve estar dentro do diretório de seu repositório
2) git pull origin (pronto seu repositório está atualizado com o remoto)

****

Comando no terminal para enviar alterações para o remoto

1) estar dentro do diretório do repositório
2) git status
3) git add .
4) git commit -m 'aqui vc informa suas alterações em forma de texto'
5) git push origin master (após esse comando, talvez o git peça seu login (usuário e senha) do gitHub)
6) caso não ocorra nenhuma mensagem de "error", suas alterações foram para o repositório remoto (gitHub).

****

Comando no terminal para inicializar um repositório qualquer

1) estar dentro do diretório que deseja versionar pelo git (cd)

2) git init (será criada uma pasta .git dentro do diretório)

3) git status (mostrar os arquivos que estão lá)

4) git add . (ou um add por arquivo, informando o nome do arquivo)

5) git status para ver o estado atual do repositório

Caso não tenha ainda configurado o email e nome de usuário no git, faça os seguintes comandos:

a) git config user.email "nomeDoUsuario@gmail.com"

b) git config user.name "nome do usuario aqui"

6) git commit -m 'aqui vc informa uma mensagem em forma de texto'

7) git log (para ver informações do que vc fez)

8) Para enviar esse repositório para o gitHub, siga os passos abaixo

9) Faça login em seu gitHub

10) crie um repositório e copia a url

11) de volta ao terminal

12) git remote add origin + a url

13) git remote -v (visualizar o repositório clonado)

14) git push -u origin master (talvez pedirá usuário e senha do gitHub)

15) vá no gitHub, refresh a página e verá que seus arquivos foram para lá.

Curso de git e GitHub – Rastreando e recuperando versões anteriores do projeto(checkout)

Dentro do diretório no terminal demos o comando:

$ git log ou resumir a informação principal do commit pode executar o comando

$ git log –oneline

Quando dermos o comando git log, o último hash do último commit sempre trará a expressão HEAD – MASTER, indicando que o projeto está em sua última versão dentro do ramo master

$ git branch

$ git log –oneline – visualizo todos os commits realizados

$ git checkout + os primeiros 7 caracteres do hash do commit desejado, me retornará a versão daquele commit

$ git log –oneline – após o comando acima (git checkout + hash do commit), o HEAD agora passa a apontar para a versão referente ao commit do checkout que fizemos

$ git checkout master

$ git log --oneline

$ git branch

$ git diff – para visualizar as diferenças commitadas

$ git status

$ git reset ––hard + o último hash (volta para uma versão anterior)

Criando branch (ramificações do projeto)

https://www.youtube.com/watch?v=iRs6sQOPcvg

git branch

git log –oneline

git checkout -b teste (para criar uma nova branch e já estar logada nela)

git branch + "novo nome da branch" (apenas cria uma nova branch)

git log –oneline

git branch + "o nome da branch" (para entrar no ramo)

git branch -d + "o nome da branch" (para remove/excluir uma branch)

git branch -m + "novo nome" (para renomear um branch)

git log ––oneline ––grapf > para visualizar o gráfico de commits

git log ––oneline ––grapf --all

git log –graph –all

Entendimento dos tutoriais: Quando inicializamos um repositório, por default é criado o ramo master (branch)

Trabalhe à vontade e faça as alterações dentro do ramo master e salve.

Crie uma nova branch (git checkout -b + "o nome da nova branch")

Faça um merge com a branch "master", desta forma a branch "auxiliar" estará atualizado com a branch "master"

Ao fazer o push para o repositório remoto, use a branch "auxiliar", como também o pull

Desta forma o ramo "auxiliar" ficará atualizado com o remoto, e depois faça um merge com o ramo "master", que por sua vez também ficará atualizado.

Obs. importante: mesmo depois que fazer um merge entre branch's deve fazer um add e um commit com a mensagem específica do que foi feito neste merge.

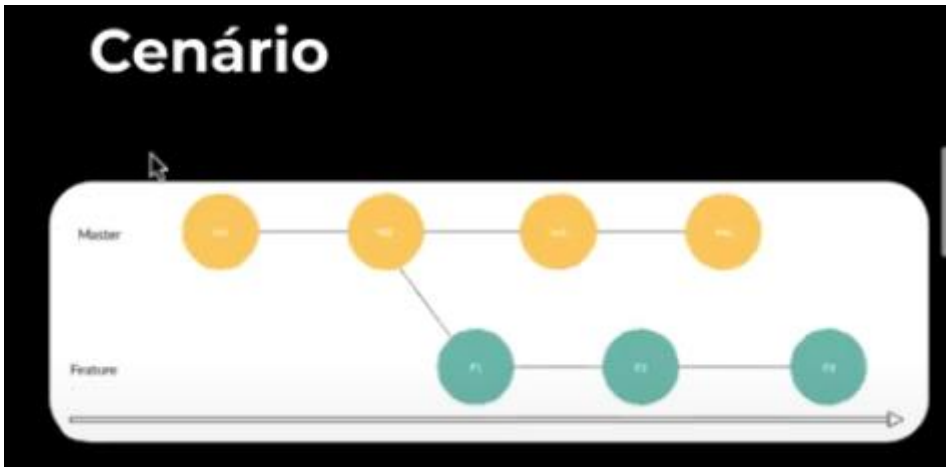
git revert --no-edit 222cccc : Se omitirmos a opção --no-edit, será aberto um editor de texto para editarmos a mensagem do novo commit.

Trabalhando com Branch's e diferenciando Merge, Squash e Rebase

https://www.youtube.com/watch?v=8znk5eDbVGA

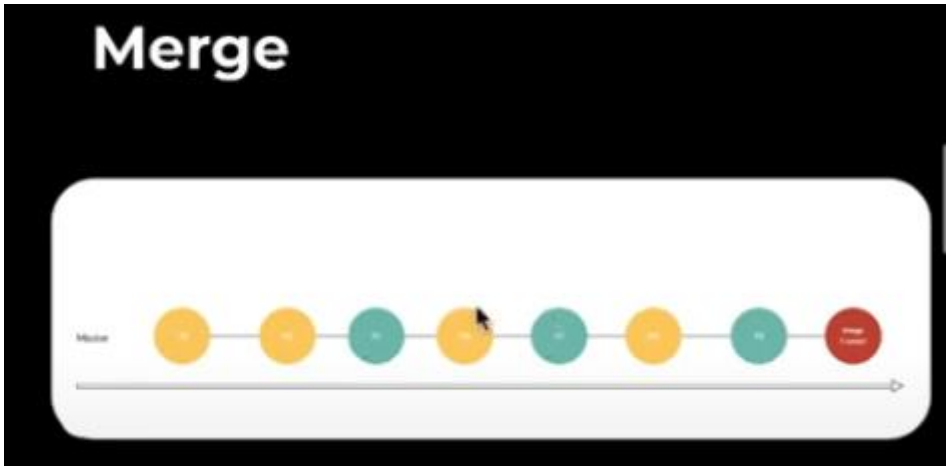
prof. Angelo Luz

Cenário sem merge, ilustrando as duas branch's em seu ambiente separado



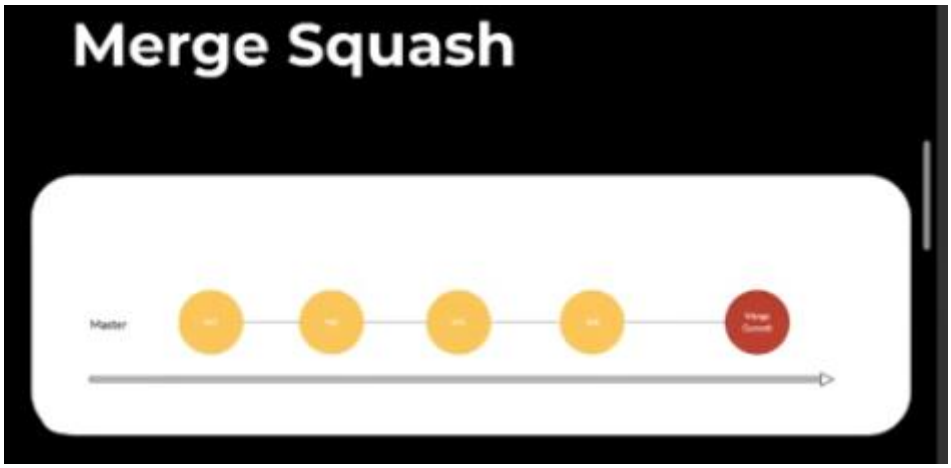
---

git merge dev – imagem abaixo ilustra a mesclagem dos arquivos da branch dev com a branch master, observe que a mesclagem ficou de acordo com a linha do tempo de cada commit realizado em cada uma das branch's.



---

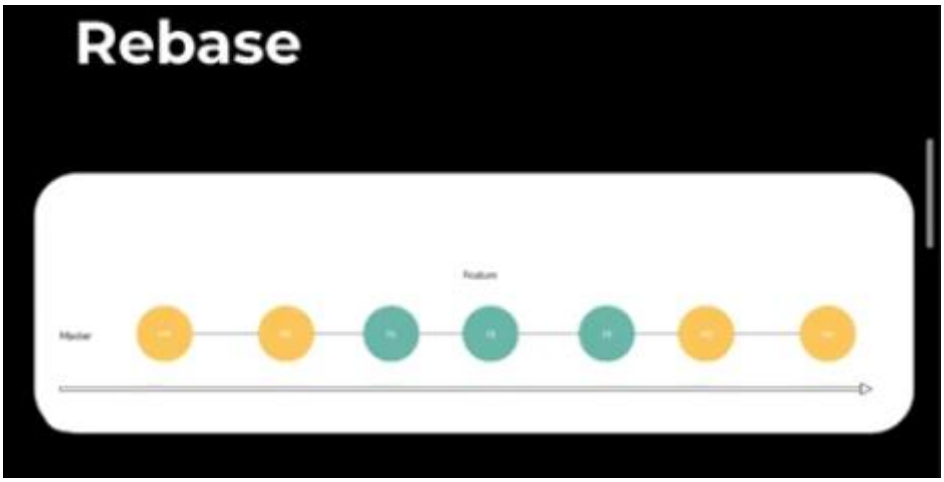
git merge dev –squash – esse tipo de merge traz os arquivos da branch dev, porém não finaliza, os arquivos trazidos ficam na stage área, adicionados, esperando para serem comitados.



Observe que na imagem acima que, o squash agrega os arquivos a branch master, porém não de forma mesclada, gerando apenas um único commit.

Vantagem: o log das branch's ficam menos poluídos.

Git rebase dev - neste caso, o rebase não cria um commit novo. Para entender melhor, demos o seguinte comando: git log --oneline, percebemos que os arquivos ficaram entre um e outro commit da master, conforme ilustra a imagem abaixo:



Com esse tipo de merge não é possível usar o recurso reset, por isso o rebase deve ser usado com muita cautela, somente alguns casos específicos. Segundo orientação do tutorial, o rebase é mais indicado quando for fazer merge no sentido contrário, da master para o dev. 99,9% dos merges devem ser feitos com o comando "merge".

Dica: git commit -m 'msg' –amend > para renomear a mensagem de um commit já realizado

para reverter/desfazer um commit:

git reset HEAD~1 –hard > este comando desfaz um commit e remove os arquivos da branch na qual vc está logado

Escondendo arquivos para navegar entre as branch's

https://www.youtube.com/watch?v=jisXznpf7AQ

prof. Angelo Luz

usando o stach:

git stash > para tirar um determinado arquivo da stage área

git stash list > lista os arquivos que estão no stash

git stash pop > desempilha o arquivo e devolve para a stage área

git stash save 'cria um rótulo para melhor identificação'

git stash apply 1 > devolver o arquivo na stage área

git stash drop 1,2.... > apaga o arquivo

git stash show > visualizar

git stash clear > apaga tudo

Repositórios remotos (push, pull x fetch, clone, reflog e cherry-pick)

https://www.youtube.com/watch?v=PomkBfARP_E

prof. Angelo Luz

SSH/KEY > configurar

git remote add origin

git remote

git remote show origin

git push origin master ou git push -u origin --all para enviar todas as branch'salve

após utilizar a expressão -u, os próximos push podem ser apenas: git push

git push -f (--force) para forçar um push > ou seja, este comando ignora diferenças entre repositórios, menos a branch master

git reflog > para recuperar o histórico totalmente apagado e me fornece a hash

git cherry-pick + os primeiros 7 caracteres da hash de um determinado commit

git push

git pull > baixa o conteúdo do repositorio remoto sem validar o conteudo

git fetch > baixa o conteúdo do remoto, mas sem mesclar, ou seja, primeiro verifico o que está vindo para depois de fato mesclar

git checkout origin/master > para entrar no remoto e ver a alteração/conteudo que está vindo para poder validar

git checkout master ou dev > para retornar a branch do repositorio local

git merge origin/master > para mesclar o conteúdo do remoto com meu repositorio local

git clone + url > para clonar um projeto do remoto de outra pessoa

---