Comandos Git

Fluxo de trabalho comum
Iniciar repositório: git init
Adicionar um repositório remoto: git remote add nomeAliasDadoAoRepositorio endereço (HTTPS ou SSH)
Trazer as modificações do github para o projeto local: git pull
Adicionar/propor arquivo ao git: git add nomeArquivo
Realizar commit: git commit -m "comentário"
Enviar arquivos para o repositório no Git:

• Quando é a primeira vez, utiliza o paramêtro **u** para que o Git 'guarde' a informação, então: git push -u paraOndeVai deOndeVem
• Nas próximas vezes: git push repositório nomeBranch
• Exemplo:
  • git push origin master
  • *ou*
  • git push (caso já esteja na branch desejada)

Branch
Criar uma branch: git checkout -b nomeBranch
Trocar de branch: git checkout nomeBranch
Deletar uma branch **local**: git branch -D nomeBranch
Deletar uma branch **remota**: git push origin --delete nomeBranch
Renomear uma branch: git **branch** -m novoNomeBranch

Repositório
Adicionar um repositório remoto: git remote add nomeAliasDadoAoRepositorio endereço (HTTPS ou SSH)
Clonar (obter) um repositório: git clone "url"
Forkar (copiar) um repositório: está etapa é feita diretamente no site do GitHub, no projeto.
Visualizar repositórios adicionados: git remote -v
Merge develop com branch atual: git fetch origin && git checkout develop && git pull origin develop && git merge origin/develop

---

Relatórios
Reportar como o diretório (leia repositório na "linguagem git") **está** no momento: git status
Visualizar o histórico (hash/identificador, autor da modificação, data e mensagem) dos *commits* realizados: git log
Visualizar as mudanças antes de commitar: git diff

---------- **MAIS USADOS**

Resets - **cuidado**
Resetar modificações:
Resetar modificação, retornar arquivo para antes da edição **anterior à proposta** (git add) do arquivo ao Git: git checkout --nomeArquivo
Resetar modificação, retornar arquivo para antes da edição **posterior à proposta** *staged* (git add) do arquivo ao Git: git reset nomeArquivo ; git checkout nomeArquivo

Resetar commits:
Resetar commit, retornar arquivo para a ***staged*** (git commit): git reset --soft
Resetar último commit, retornar arquivo para a ***staged*** (git commit): git reset --soft HEAD^ **<----**
Resetar commit antes de fazer o *push*, mantendo as modificações: git reset HEAD~1 --soft
Corrigir/refazer último commit: git commit --amend

Resetar commit, resetar **modificações** (local e/ou repositório) no arquivo *1*: git reset --hard <old-commit-id>
Resetar commit, resetar **modificações** (local e/ou repositório) no arquivo *2*: git push -f <remote-name> <branch-name>
Recuperar git reset -HARD -> https://stackoverflow.com/questions/5788037/recover-from-git-reset-hard

Resetar add:
Resetar add (inclusão) **não mantendo** alterações de **um item**: git reset <nomearquivo>
Resetar add (inclusão) **não mantendo** alterações de **todos** os **itens**: git reset
Resetar add (inclusão) mantendo alterações, retornar arquivo para **antes da *staged*** (git add): git reset --mixed

Resetar modificações:
Resetar modificação de **um item**: git checkout <nomearquivo>
Resetar modificação de **todos** os **itens**: git checkout .

DICA
Sabendo nome id do commit: git log

---------- **FIM**

Guardar alterações
Guardar alterações: git stash
Restaurar alterações: git stash apply

Merge outra branch na branch atual
```
$ git checkout branchCorreta
$ git pull origin
$ git merge origin/develop
$ git pull origin
$ git commit

ou... ver "Merge develop com branch develop" acima
```

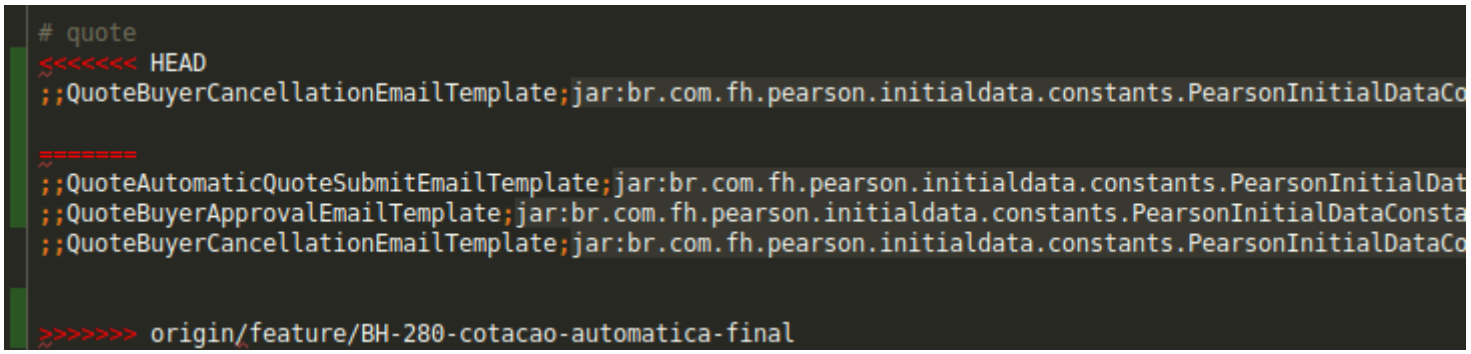
Resolver conflitos

Passo a passo:
```
$ git checkout nomeBranchAPartirDeOndeSeraCriadoANovaBranch
$ git pull origin nomeBranchAPartirDeOndeSeraCriadoANovaBranch
$ git checkout nomeBranchNova
$ git pull origin nomeBranchNova
$ git st
$ git merge origin/nomeBranchAPartirDeOndeSeraCriadoANovaBranch
$ git push origin nomeBranchNova
```

Exemplo develop:
```
$ git checkout develop
$ git pull origin develop
$ git checkout branchName
$ git pull origin branchName
$ git st
$ git merge origin/develop
$ git push origin branchName
```

DICA: Resolvendo conflitos...
1 - Git diff
1. $ git diff nomeArquivo
2. Procurar o que tá dentro do HEAD
3. Comparar o que tá dentro do “<<<<<<<” da minha branch
4. Se tiver tudo ok, não repetiu nenhum código
   4.1. Apagar as linhas "<<<<<<<" que o git coloca
5. E foi

Leitura de conflito:
O que está dentro do “<<<<<<< HEAD ======” é o que está na branch atual. O que está abaixo de “ ======” até “>>>>>>>” é o que está na branch que quer fazer o merge



2 - GitKraken
O GitKraken considera duas branchs na hora de resolver o conflito, como sendo:
Branch **A** - branch que você está
Branch **B** - branch que você está tentando mergear dentro da A

Na branch A é a junção entre os dados modificados na branch B com o que já tinha na branch A. Ou seja, o desenvolvimento realizado que está na branch B, já está automaticamente dentro da A, na maioria das vezes **basta escolher a A para resolver o conflito**. Somente atente-se a conferir se realmente se a branch A contempla a branch B.

Confirações extras

Visualizar/listar conflitos: git diff --name-only --diff-filter=U

Trocar editor padrão de commit do git: git config --global core.editor "vim"

Recuperar stash perdido:

Procurando o hash do stash:
- Visualizar pelo software gitk: $ gitk --all $(git fsck --no-reflog | awk '/dangling commit/ {print $3}')
- Visualizar pelo terminal: $ git log --graph --oneline --decorate --all $(git fsck --no-reflog | awk '/dangling commit/ {print $3}')

Cada hash de cada stash virá com um *WIP on* anterior ao seu nome, sendo assim procure por WIP on...

Aplicar o stash pelo hash: $ git stash apply stash_hash

Referências e mais...
https://gist.github.com/leocomelli/2545add34e4fec21ec16
https://stackoverflow.com/questions/3065650/whats-the-simplest-way-to-list-conflicted-files-in-git
https://stackoverflow.com/questions/2596805/how-do-i-make-git-use-the-editor-of-my-choice-for-commits
https://stackoverflow.com/questions/89332/how-to-recover-a-dropped-stash-in-git