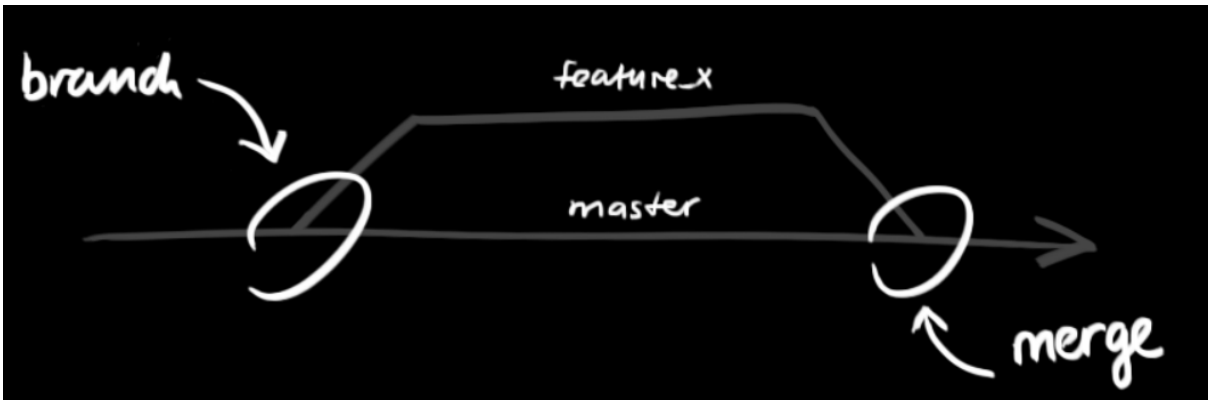
Guia Prático de Utilização Git/GitHub

Douglas Silva da Silva
Práticas de Engenharia de Software
2021-2

Primeiramente, devemos distinguir Git de GitHub. Git é o **sistema de gerenciamento de versão**, gratuito e de código aberto, criado por Linus Torvald (por curiosidade, o mesmo criador do sistema operacional Linux) em 2005.
Já o GitHub é uma empresa, adquirida recentemente pela Microsoft por U$7,5 bilhões, que **desenvolve soluções baseadas no Git**. Seu produto mais conhecido é o GitHub.com, que é uma plataforma que une o gerenciamento e hospedagem de código-fonte a funções de redes sociais, como feed, comunidades, fóruns, etc.
Após ter criado uma conta no GitHub e instalado o Git na sua máquina, já é possível hospedar projetos no GitHub.com. Para isso, é necessário seguir os seguintes passos:

1. Após abrir o Prompt de Comando, localize a pasta do projeto e digite **git init** para criar a estrutura básica do repositório.
2. Para obter um repositório, crie uma cópia do trabalho em um repositório local, executando o comando **git clone /caminho/para/o/repositório**. Para servidores remotos, o comando será o **git clone usuário@servidor:/caminho/para/o/repositório**.
3. É possível propor mudanças (adicioná-las ao Index) usando o comando **git add <arquivo>** para um arquivo em específico, ou **git add *** para adicionar todos arquivos do local. Para realmente confirmar estas mudanças, é realizado o que chamamos de **commit**, através do código **git commit -m "comentários das alterações"**. Com isso, o arquivo é enviado para o HEAD, mas ainda não para o repositório.
4. Para enviar estas alterações ao seu repositório remoto, é executado o comando **git push origin master**.

Branches (ou ramificações) são utilizados para desenvolver funcionalidades isoladas umas das outras. O **branch master** é o branch “padrão” quando é criado um repositório. É utilizada outras branches para desenvolver e mesclá-los (ou realizar o **merge**) à branch master após sua conclusão:



Para criar uma nova branch, chamada “funcionalidade_x” por exemplo, é utilizado o comando **git checkout -b funcionalidade_x**. Para retornar para a master, usa-se **git checkout master**. E para remover o branch criado, é utilizado o código **git branch -d funcionalidade_x**.
Uma branch não está disponível a outros usuários a menos que você envie a branch para seu repositório remoto, pelo comando **git push origin <funcionalidade_x>.**

Para atualizar repositórios locais com a mais nova versão, é executado o comando **git pull** na sua pasta de trabalho, para obter e realizar merges remotas.
Para realizar merges de uma outra branch para seu branch ativo (master, por exemplo), é usado o **git merge <branch>**.
Em ambos os casos o git tenta fazer o merge das alterações automaticamente. Porém, nem sempre é possível e resulta em conflitos. Cada usuário é responsável por fazer o merge destes conflitos manualmente, editando os arquivos exibidos pelo git.
Após alterar, é preciso marcá-los como merged, com o **git add <arquivo>**, pré-visualizando as alterações com o comando **git diff <branch origem> <branch destino>**.

Em caso de erros, no qual é necessário ajustes, você pode subscrever as alterações locais usando o comando **git checkout -- <arquivo>**. Este comando substitui as alterações na sua árvore de trabalho com o conteúdo mais recente no HEAD. Alterações já adicionadas ao index, bem como novos arquivos serão mantidos.
Se ao invés disso você desejar remover todas as alterações e commits locais, recupere o histórico mais recente do servidor e aponte para seu branch master local com os comandos **git fetch origin** e **git reset --hard origin/master.**