COMANDOS GERAIS GIT

git init (iniciar repositório git no diretório)

git commit (comitar arquivos)

rm (remover arquivos)

rm testeFatorialRecursivo.* (remove todos os tipos de arquivos com o mesmo nome)

ls (consultar a lista de arquivos dentro do diretório)

git help - Exibe informações de ajuda sobre o Git

git init - Cria um repositório do Git a partir do diretório que está

git clean –n – Mostra os arquivos que serão removidos e não foram comitados

git clean –f – Remove os arquivos que não foram comitados. Limpa a área de trabalho.

git reset --hard ijek2l3j4l (número de hash) – desfazendo commit

git log - Mostra logs de confirmação

git status - Mostra o status da árvore de trabalho

git ls - Mostra informações sobre arquivos no índice e na árvore de trabalho

git rm – Remove arquivos do diretório da área de trabalho
rm testeFatorialRecursivo.* (remove todos os tipos de arquivos com o mesmo nome)

git commit –m <mensagem do commit> - comita arquivos

git commit --amend – Alterar a mensagem do último commit

git --version – verifica a atual versão do git instalada no computador

pwd – verifica em qual diretório estou

ls –la – verifica arquivos ocultos no diretório atual

git config --global user.name “Heloisa Felizardo Campos” – configurando usuário global

git config --global user.email “helocmp@yahoo.com.br” – configurando email do usuário global

git config --global core.editor “vscode.exe” – configurando editor de texto usado para editar o git

git config --global color.ui true – configurando a cor das mensagens de commit mais visíveis

git config --list – verifica quais são as configurações disponíveis

git config --global init.defaultbranch "main" – Configurar o nome da branch principal

git config <init.defaultbranch> - consultar as configurações gerais

touch <nome do arquivo> - Criar arquivos (Via Git bash)

git mv testeFatorial1 testeFatorialRenomeado1.java – criar pasta/renomear ou mover arquivos

mkdir <teste> – criar diretórios

git checkout -- projetos/aplicacao-juros/juros.js – Desfaz alterações anteriores no arquivo na área de trabalho

cat projetos/aplicacao-juros/juros.js – visualizar o conteúdo do arquivo

git restore --staged <file>..." to unstage area para área de trabalho

git commit --amend -m "Arquivo para simular a funcionalidade de desfazer o último commit"

git clean –n – verifica o que vai ser removido fora da stage area

git clean –f – força a remoção dos arquivos que ainda não estão na stage area

git reset --soft ijek2l3j4l (número de hash) – desfazendo commit atual para um commit anterior

git reset --hard ijek2l3j4l (número de hash) – desfazendo commit

git restore --stage <arquivo ou pasta> remove os arquivos do stage

git commit –am “mensagem” – manda o arquivo para stage e comita ao mesmo tempo.

cat .git/refs/heads/main – comando de referência do último commit no main

git log --oneline -3 – mostra as últimias 3 linhas por commit. Resumos dos commits. Utilizado para logs muito grandes

git show HEAD – Mostra quais foram as alterações feitas no último commit

git log --graph --oneline --decorate --all – vai exibir o topo de cada branch, para onde o conteúdo head está apontando em cada branch

git diff main..fatorial-recursivo – compara o código que foi alterado entre duas branchs

git branch --merged – usado para mostrar todas as branchs que estão incluídas dentro da branch atual na qual você está dando commit

git branch --move – para renomear o nome da branch

git branch -d <branch-para-remocao> – remove uma branch

git branch --delete <branch-para-remocao> – Deleta uma branch

git branch –D <branch-para-remocao> - Deleta definitivamente uma branch

git merge <nome da branch> - unificar os arquivos dentro da branch main

git merge --abort – Aborta o merge em caso de conflito no merge

Como evitar conflitos de merge

- Fazer merge constantes e evitar fazer commits com grandes alterações.
- Vai alterando e fazendo commit.
- Manter linhas mais curtas o possível.
- Fazer commits pequenos e bem focados.
- Evitar alterar espaços em branco, tabulações, etc.
- Fazer merges constantes.
- Não precisa eliminar a branch depois de fazer o merge
- Fazer as branchs estarem alinhadas com a main. Sincronizadas com a main.

git push – enviar dados para o repositório remoto

git fetch origin – sincronizar repositório remoto para o local

Obs.: Antes de fazer o push faça o fetch para atualizar sempre o repositório local. Fazer o fetch antes de qualquer coisa e frequentemente.

git merge origin/main main – Faz merge dos dados remotos com os dados locais

git remote add origin – adiciona o repositório remoto

git remote rm origin – remover o repositório remoto

git push –u origin main – fazer upload dos dados no repositório remoto na branch main (se não usar –u o sincronismo com a branch remota não é feita)

cat .git/config – verificar as configurações atuais no git local e remoto

git branch –r – exibe somente as branch remotas

git branch –a – exibe todas as branchs remotas e locais

git branch –d sem-sincronismo – remove branch local “sem-sincronismo”

git clone https://github.com/HeloisaFelizardo/ProjetoFatorial.git ProjetoFatorialFulano - Clona um repositório remoto e já cria um diretório renomeando o nome da pasta de destino

cat .git/config – para saber se os dados locais estão sincronizados com os dados remotos

git pull = git fetch + git merge

git branch sem-sincronismo origin/sem-sincronismo – para rastrear a branch “sem-sincronismo” remota na branch “sem-sincronismo” local

git checkout -b sem-sincronismo origin/main - cria branch “sem-sincronismo” e sincroniza com a branch remota

Obs.: Sempre fazer primeiro os comandos git fetch (faz o download dos dados remotos para o local) - git merge (sincroniza os dados local e remoto) - git push (envia para o servidor remoto as alterações feitas no local) do servidor remoto antes de qualquer coisa

git push origin :sem-sincronismo - remove a branch remota “sem-sincronismo”

git add –i - para selecionar quais arquivos ou partes de arquivos serão armazenados no stage