**COMANDOS DO GIT**

**CHAVE SSH**

ssh-keygen -t rsa -b 4096 -C [seu-email@seu-servidor.com](mailto:seu-email@seu-servidor.com)

Digitar a senha 2 vezes

**Utilizar o comando abaixo para copiar a chave e adicioná-la ao github**

clip < ~/.ssh/id\_rsa.pub

**LISTAGEM**

**ll ou ls -la =** lista os arquivos do diretório

**git init –bare =** inicia um novo repositório git de forma local

**LOG**

**git log --name-status =** mostra o log dos commits, juntamente com o status dos arquivos que foram modificados, adicionados ou deletados

**git log --pretty=oneline =** mostra os commits em linha

**git log --abrev-commit =** mostra os commits abreviados

**git log --pretty=oneline --abbrev-commit =** mostra o commit abreviado com uma linha

**git log --stat** = mostra alguns dados estatísticos

**git log -p =** mostra as alterações de cada commit e por quem foi alterado

**git log -p -3=** mostra as alterações dos 3 últimos commits

**git log --pretty=format:"%h - %an, %ar : %s" =** mostra a hash abreviada , o autor-name, o tempo ou data e o título do commit

**DIFF**

**git diff --staged =** mostra o que foi modificado no nível de staged

**git diff <numero commit> <numero commit> =** compara 2 commits que podem ser com números abreviados

**git diff -w =** mostra as alterações apenas de texto e outros sem os espaços

**COMMIT**

**git commit -m "mensagem" --amend =** modifica o texto do último commit e também substitui o mesmo. Tomar cuidado de não ter feito **push** antes, page evitar conflitos

**MERGE**

**git merge --abort =** o git tenta abortar um merge que entra em conflito

**git mergetool =** abre a ferramenta configurada pra fazer o merge

**REMOÇÃO**

**git rm arquivo =** deleta o arquivo na staging area

**git rm -rf node\_modules/ --cached =** remove o node\_modules da staging area quando adicionado acidentalmente, sem deletar o diretório real

**rm node\_modules -rf =** remove a node-modules do working-directory, juntamente com o seu diretório

**CONFIG**

**git config --list =** mostra configurações padrão do git

**git config -l =** mostra as configurações principais do git

**git config --global core.editor =** mostra o editor configurado para fazer commits

**git config --global core.editor "code --wait" =** abre o VS code para edição, ao fazer um git commit

**git config --global -e** = modifica manualmente as configurações do git

**git config --global merge.tool meld =** muda o editor de conflitos de vim para meld

**git config --global difftool.meld.path "/c/Program Files (x86)/meld/meld.exe" =** faz o diff no meld

**git config --global mergetool.meld.path "/c/Program Files (x86)/meld/meld.exe" =** configura o meld no Windows

**echo "\*.orig" >> .gitignore =** acrescenta no gitignore, os arquivos gerados automaticamente pelo meld

**BRANCH**

**git branch -m <nome-da-branch>** = renomeia a branch

**git checkout --orphan <nome-da-branch>** = Cria uma branch nova sem pegar dados do histórico de outras branchs, ou seja, vazia.

**MERGE**

**git rm -rf =** limpa todos os arquivos da arvore do git. Vantajoso para trabalhar numa branch vazia, onde precisa começar os arquivos do zero

**REPOSITÓRIO - REMOTE - CLONE**

**git init --bare =** cria um repositório remoto local (principal)

**pwd =** retorna o caminho completo de um diretório no computador

**git remote add <origin> < caminho do repository> =** adiciona um repositório remoto no projeto

**git remove -v =** mostra o repositório de origem (fetch) e a origem (push)

**git clone <caminho do repository> =** clona um repositório existente

**git remote rm <origin> =** remove um repositório remoto

**git remote set-url <origin> <url> =** Altera a url remota de um repositório

**CHECKOUT**

**git checkout .gitignore OU outro arquivo =** retira o mesmo da working directory, voltando as alterações ainda não comitadas

**git checkout <numero-do-commit> =** Volta o commit no tempo e permite alterações para ser mergeada com outra branch. É obrigatório criar uma branch a partir deste

**git checkout <branch> -b <nome-da-branch> =** cria uma nova branch baseada na branch específica. Exemplo, se está na branch master e precisa criar uma uma nova branch a partir da dev. Pode ser usada também com um commit voltado no tempo, colocando o número do commit ao invés do nome da branch

**STASH**

**git stash =** congela o working directory com as alterações que ainda não foram comitadas, para trabalhar em outra branch

**git stash list =** lista os commits e a alterações que estava sendo feita antes do git stash retornando um **stash@{numero}**

**git stash apply =** retorna o working directory ao estado das últimas alterações, para finalmente serem comitadas

**git stash drop stash@{numero} =** remove o stash

git stash pop = aplica o ultimo stash e dropa o penúltimo