Tutorial Git

git config --global user.name "caio" //configurando o nome de usuario

git config --global user.email "caio@gmail.com" //configurando o email

git config --global user.password "123456" //registrando a senha

git config --global core.editor <seu editor de texto> //vim sublime emacs

git config --global core.editor //se nao for definido o editor por padrao sera vim

git config user.name //exibe o nome

git config user.email //exibe o email

git config --list //exibira todas as informações

File Status Lifecycle: untracked - unmodified - modified - staged

git add <file>

git commit -m "sua mensagem"

git commit -m "sua mensagem" <nome de um arquivo específico>

git log //exibe o log

git log --decorate //exibe o log

git log author="nome do autor" //exibe o log a partir de um nome de usuario especifico

git shortlog //exibe o log com os autores e a quantidade de commits e as mensagens dos commits

git shortlog -sn //exibe o nome dos atores e suas respectivas quantidades de commits

git log --graph //exibe em forma grafica

git show <numero da hash> //o numero da hash pode ser verificado com git log

git diff //exibe as modificaçoes nos arquivos

git diff <nome do arquivo> //exibe as modificaçoes em um arquivo especifico

git diff --name-only   //exibe o nome dos arquivos modificados

git checkout arquivo //para voltar antes da edicao - nesta situacao o arquivo estara como: modified voltando para unmodified

Voltando com o arquivo para um ponto anterior ao staged área

Nesta situação o arquivo estará como:staged.

git add arquivo //jogando na staged área
git reset HEAD arquivo //tira o arquivo da staged área - o arquivo ficará agora com o status: modified
git checkout arquivo //agora de acordo com os dois passos anteriores eu vou para um ponto anterior a mudança, ou seja, status: unmodified

Voltando com o arquivo para um ponto anterior ao do commit

Nesta situação o arquivo encontra-se commited.

git add arquivo //jogando na staged area
git commit -m "comitando o arquivo"

git reset --soft <hash> ==> mata o commit e volta para o status staged
git reset --mixed <hash> ==> mata o commit e volta para o status modified
git reset --hard <hash> ==> mata o commit e volta para o status unmodified

O hash pode ser visto com "git log".

OBS.: Na fase unmodified o arquivo já pode ser comitado, pois este status é considerado pronto pelo git para commit.

Listar todas as configurações do Git:

git config - - list

Apagar variavel da lista de configuraçoes do Git:

git config --global --unset <nome da variável>

Ligando o repositório local ao remoto (o comando abaixo pode ser obtido no própro GitHub):

git remote add origin git@github.com:Marquedante/nome_do_repositorio.git

origin => nome fornecido ao repositório remoto por padrão

Exibindo repositório(s) remoto(s)

git remote

Exibindo repositório(s) remoto(s) com endereço completo

git remote -v

Enviando arquivos para o repositório remoto:

git push -u origin master

origin => repositório remoto - origin é um nome padrão, mas poderia ser qualquer outro.
master => nome da branch no repositório remoto - você pode ter mais de uma branch.

Clonando um repositório:

git clone <endereço do repositório remoto a ser copiado> <nome desejado do repositório - opcional>

==========

Removendo proxy errado do Git:

git config --global --unset http.proxy
git config --global http.sslVerify false

//git config --global --unset https.proxy
//git config --global https.sslVerify false
==========

Inserindo proxy no Git:

git config --global http.proxy http://username:password@proxiURL:proxiPort
//git config --global https.proxy http://username:password@proxiURL:proxiPort

git config --global http.proxy http://username:senha@proxy_da_rede:PORTA

git config --global http.proxy http://marcosmarques:senha@proxy.tj.rj.gov.br:80

Alterando a senha no Git:

1) Verifique as credencias no Windows: clique no botao iniciar do windows a seguir persquise por cofre em seguida remova as credencias pois provavelmente sao antigas.
Não se preocupe, pois quando for inserida a nova senha do git através do prompt, Git Bash, será criada automaticamente uma nova credencial no Windows.

2) No Git Bash faça:

>git config --global http.proxy http://username:password@proxiURL:proxiPort
//git config --global https.proxy http://username:password@proxiURL:proxiPort

git config --global http.proxy http://username:senha@proxy_da_rede:PORTA

git config --global http.proxy http://marcosmarques:senha@proxy.tj.rj.gov.br:80

git config --global https.proxy https://marcosmarques:senha@proxy.tj.rj.gov.br:80

Apagar as configurações de proxy do Git ou alguma outra configuração:

git config --global --unset http.proxy

Criando uma Branch

git checkout -b <nome da nova branch>

Listando as Branches

git branch

Mudando de Branch

git checkout <nome da branch>

Enviando a branch para o Servidor Remoto

git push <nome do repositorio remoto por padrao: origin> <nome da branch que queremos enviar>

Deletando uma branch local

git branch -D <nome da branch>

Deletando uma branch remota

git push <repositório remoto – por padrão origin> --delete <nome da branch a ser deletada>

Unindo Branches – Merge

git merge <branch principal> <branch com a qual se quer fazer o merge>

Obs.: Merge é uma forma de unir branches, onde se cria um commit extra, o qual não terá arquivo nenhum, apenas para juntar os branches. Nesta forma de união de branches, os commits não alinham-se de forma linear gerando o que chamam de diamantes.

Unindo Branches – Rebase

git rebase <branch principal> <branch com a qual se quer fazer o merge>

Obs.: Rebase é um tipo de merge onde não criamos um commit extra e utilizamos a técnica do fastforward para juntar os branches. Nesta forma de união de branches, os commits alinham-se de forma linear alocando o commit para o início da fila.

Configurando o arquivo .gitignore

1) Ignorando arquivos de determinada extensão:

1.1) Abra o arquivo .gitignore e insira a extensão do tipo do arquivo a ser ignorada:
*.zip //ignorando arquivos de extensão .zip
*.xpto //ignorando arquivos de extensão .xpto
*.class //ignorando arquivos de extensão .class

2) Ignorando um arquivo ou arquivos específicos de determinada extensão:

2.1) Abra o arquivo .gitignore e insira a extensão do tipo do arquivo a ser ignorada:
conf.zip //ignorando o arquivo conf.zip
abc.ptk //ignorando o arquivo abc.ptk
leia-me.txt //ignorando o arquivo leia-me.txt

3) Arquivo **.gitignore** padronizado para projetos:

fonte: https://github.com/github/gitignore

Salvando arquivos na Área de Stash

git stash save <nome do arquivo>

Listando arquivos salvos na Área de Stash

git stash list

Recuperando arquivo(s) da Área de Stash

git stash apply <nome do id do stash – pode ser identificado com stash list>

Realiza a remoção de todos os stash's

git stash clear

Ambos criam uma mensagem/nome para o stash

git stash create <mensagem> *ou*

git stash save <mensagem>

Criando Tag: como marcador de uma release

git tag -a <número da versão desejada> -m “mensagem para sua tag criada”

Exemplo: git tag -a ***1.0.0*** -m “mensagem para sua tag criada”

Enviando a Tag para o servidor remoto

git push <servidor remoto> <nome da branch> --tags

Listar Tags

git tag

Deletando Tags do repositório local

git tag -d <versão da tag>

Deletando Tags do repositório remoto

git push <servidor remoto – por padrão origin> **:**<versão da tag>

Revertendo Commits

Pode acontecer o caso de você precisar desfazer um commit que não é o commit mais recente e sim um mais antigo. Nesse caso, fazer um reset até o commit irá apagar também todos os commits na frente dele. Para resolver isso temos o revert. O revert cria um novo commit que faz o reverso do commit especificado. Ou seja, se o commit adicionou um arquivo, o revert remove, se editou uma linha, volta ao que era antes. Um outro ponto importante é que o revert, ao contrário do reset, não apaga commit. Supondo que queremos reverter o commit de ID 11a5b usamos:

git revert <número do commit>

==========

Comitando pelo Visual Code:

1. Commitar tudo o que fora feito sem push.

2. Fazer pull.

3. Se houve conflito, corrigir aceitando mudanças novas e preservando as atuais se precisar e POR FIM COMMITAR.

4. Subir a aplicação.

5 Se a aplicação subiu sem erros faça: push

Observação: se houve conflito será necessário commitar tudo após resolver os conflitos e em seguida faça push.

Instalando as dependências através do npm

npm install -g

Instalando o Angular Cli com npm

npm install -g @angular/cli //pegará a última versão – instalação global

npm install @angular/cli //pegará a última versão – instalação local

npm install @angular/cli@xx.xx.xx //instalando uma versão específica

Configurando Proxy no Node

npm config set proxy http://nome_usuario:senha@meu.endereco.proxy:80

npm config set https-proxy http://nome_usuario:senha@meu.endereco.proxy:80

npm config set proxy http://nome_usuario:senha@proxy.tjrj.jus.br:80

npm config set https-proxy http://nome_usuario:senha@proxy.tjrj.jus.br:80

Listando Variáveis no Node

npm config list

Removendo Variáveis no Node

npm unset nome_da_variável