Olá, meu nome é Johnny Alexandre!

Estou estudando para me tornar um Dev FullStack Java Júnior, cursando ADS na FIAP (2° Semestre) e fazendo o curso de programação do Instituto PROA

**ANOTAÇÕES DO APRENDIZADO**

**Para configurar o git, use os comandos:**

**git config --global user.email** **"**Email que foi colocado na criação da conta do git**"**

**git config --global user.name** **"**Nome do Usuário**"** - Quando enviar a versão, o nome estará atrelado a versão.

**1 - Como inicializar um repositório?**

**git init** - Cria um novo repositório no git

**git status** - Verifica o status do repositório | Verde: Pronto para ser comitado

**2 - Como adicionar arquivos ao controle de versão?**

**git add "nome-do-arquivo"** - Prepara um arquivo para ser adicionado ao controle de versão

**git add .** - Adiciona todos os arquivos que não foram adicionados

**3 - Como criar versões do código com commit**

**git commit -m "MensagemFalandoSobreOQueFoiFeito"** - Cria a primeira versão do código (o -m é de mensagem), colocar a primeira versão como "commit inicial" é uma boa prática.

**4 - Como enviar alterações usando git push**

**git push** - Envia a versão do código para o servidor da nuvem

**git remote add origin "Link do repositório no github"** - Define para onde o código será enviado

**--set-upstream origin master** - Envia o código para a master

**5 - Ciclo de atualização do código**

Os passos para a atualização de um código são:

**1 - git status** || Verificar os status dos arquivos, se foram enviados ou não.

**2 - git add .** || Adicionar os arquivos a versão deixando-os aptos para o commit

**3 - git commit -m "NomeDaATT"** || Cria uma nova versão do projeto

**4 - git push** || Envia a versão do código para o servidor da Nuvem

**6 - Como verificar histórico de atualizações**

**git reflog** - A versão 0 sempre vai ser a mais recente (Quando utilizado esse comando, vai aparecer todos os commits feitos até o momento, seu nome e id)

**7 - Como navegar entre versões do seu código**

**git reset --hard 1863fs** || Volta para a versão do código do id (o id nesse exemplo seria o 1863fs)

**8 - O que são branchs e como criar uma**

**git branch** || Exibe os branchs disponíveis

**git branch staging** || Brache criada para receber atualizações e depois ser implementada na branch principal, o nome de uma branch pode ser o nome da nova funcionalidade.

**git checkout staging** || Muda para o versionamento da branch (staging)

**git checkout -b NovaBranch master** || Cria uma nova branch com base na branch principal

**9 - Git Merge - Como unir o código**

**git pull** || Traz as atualizações do servidor para a máquina local - depois realiza o git merge

**git merge NomeDaBranch** || A branch NomeDaBranch é a branch que contém as novas atualizações, o git merge "puxa" essas atualizações.

**10 - Pull request - Um merge democrático**

Após feito o git push, clicar em "Compare & pull request" no GitHub

**11 - GitIgnore**

**touch .gitignore** || Cria um arquivo txt do .gitignore e lá você coloca o caminho dos arquivos que devem ser ignorados, ainda é necessário adicionar esse .gitignore no projeto da branch principal.

**Resumo:**

**git init**: Inicializa um novo repositório

**git add .**: Adiciona os arquivos atuais ao próximo commit

**git status**: Verificar o status atual do repositório git

**git commit -m "mensagem do commit"**: Cria um novo commit com uma mensagem

**git push**: Envia as atualizações para a nuvem na branch atualmente ativa

**git branch**: Permite listar e ver qual branch está ativa atualmente

**git checkout nome-da-branch**: Permite mudar para uma nova branch

**git checkout -b "nome da branch de origem" "nome da nova branch"**: Permite mudar e criar uma nova branch com base em outra

**git merge "branch a receber merge"**: Permite fazer o merge da branch ativa atualmente com outra branch

**git pull**: Atualiza a branch atualmente ativa

**OBSERVAÇÕES**

* Commit -- Um commit é uma versão do código desenvolvido
* Untracked file -- O arquivo não foi adicionado ao controle de versão
* O comando git add adiciona um arquivo por vez (O que não é recomendável de se fazer em um projeto com muitos arquivos)
* GitHub - Repositório de projetos baseado no Git
* Para realizar um upload de código para um repositório, primeiro é preciso configurar o link do repositório e depois definir para qual branch enviar o código
* Branch - Divisões diferentes do código que estão sendo versionadas de forma separada. (Inicialmente enviamos para a Master)
* Para enviar uma nova atualização de código a operação a ser realizada é: git status - git add . - git commit -m "NomeDaAtualização" - git push
* O comando git merge é utilizado para unir o código novo - funcionalidade - (de uma nova branch) para a branch principal

**Como unir o código novo com a branch principal?**

Exemplo:  
Antes de fazer o merge, é preciso puxar as atualizações do servidor para o servidor local para ter certeza de que se está unindo os códigos nas versões mais atuais possíveis, isso é feito com o comando git pull

1° - git pull da branch principal - Traz atualizações do servidor para a máquina atual

2° - gerar uma nova branch a partir da branch principal

3° - Trabalhar e adicionar novas funcionalidades na nova branch criada

4° - Finalizar o trabalho na branch temporária

5° - Git checkout na branch principal (Mudar para a branch que será recebida a nova funcionalidade)

6° - Git pull

7° - Mergiar (unir) o código da branch temporária com a branch principal (depois de testar)

8° - Git push da branch principal

Praticando Git através do curso:

https://www.youtube.com/watch?v=kB5e-gTAl\_s