**O que é Git, Git Bash e comandos básicos**

Ícone

Descrição gerada automaticamente

Git é um sistema de versionamento de código comumente utilizado para projetos de desenvolvimento de software. Isso significa que ele trabalha como um intermediador entre um repositório local e um remoto (como por exemplo o GitHub), onde você pode controlar e administrar as alterações e atualizações do seu código. Lembrando que Git e GitHub não são a mesma coisa !!

Fica mais fácil de compreender o que é o Git quando analisamos o fluxo de trabalho do Git.

Se você estiver com pressa, aqui está um resumo dos comandos e o que fazem:

* **git init**: inicializa um repositório local git
* **git status**: verifica o estado dos seus arquivos
* **git add <nomeDoArquivo>**: envia seu arquivo especificado para o Stage
* **git add - -all**: envia todos os arquivos para o Stage
* **git commit -m “tituloDoCommit:**envia o que está no Stage para o HEAD
* **git remote add origin urlDoRepositorio:**adiciona e indica a URL do repositório remoto em que os arquivos serão mantidos
* **git push origin master:**envia os arquivos para o repositório remoto que você especificou através da URL do comando acima
* **git checkout -b <nomeDaBranch>:**cria uma nova branch
* **git checkout <nomeDaBranch>:**alterna para a branch especificada

**Fluxo :**

De forma resumida, o Git abrange 3 camadas. O chamado “*Working Directory*”, que é onde você está codificando seu projeto, o “*Index*”, também chamado de “*Stage*”, que é uma camada onde você adiciona suas modificações de forma não permanente, ou seja, nesta camada você pode resetar as alterações que você incluiu no *Stage/Index* ou então partir para a próxima camada, a chamada de “*HEAD*”, quando você confirma suas alterações fazendo um “commit”. Porém essa ultima camada ainda não é o seu repositório remoto para o qual você deseja incrementar suas modificações. Para que o seu código chegue até o repositório remoto é necessario um comando muito simples chamado git push. Vou explicar melhor os comandos.

**Git Bash :**

Todo esse fluxo descrito acima é manipulado através do Git Bash (caso você utilize Windows). Contextualizando, o Git foi feito inicialmente para rodar em sistemas Unix, que não é o caso do Windows. Assim , foi criado o Git Bash, uma aplicação que emula um terminal Unix e permite que você desfrute dos comandos Git. O Git Bash já vem no pacote de instalação do Git para Windows.

**Comandos :**

Existem diversos comandos Git. Aqui eu vou demonstrar apenas alguns básicos porém suficientes para que você consiga criar seu repositório, e trabalhar de forma integrada com ele.

* Primeiro crie seu repositório seu local. Sua pasta onde ficará seu projeto.
* Entre na pasta e clique com o botão direito do mouse. Selecione o Git Bash:

Interface gráfica do usuário

Descrição gerada automaticamente

* Para iniciar o repositório utilize o comando **“git init”** :

Texto

Descrição gerada automaticamente

* Você pode verificar o status do repositório com o comando **“git status”**. É uma boa prática sempre conferir o status antes de iniciar com o fluxo das alterações e durante ele também :

Texto

Descrição gerada automaticamente

Se você ainda não tiver nenhum arquivo nessa pasta, crie algum para continuar com o processo. Essa é a camada Working Directory.

* A próxima camada é o Index. Para mandar o seu arquivo para o index, basta utilizar o comando “**git add nomeDoArquivo**” ou se você quiser enviar ao Index todos os arquivos basta utilizar o parâmetro all. Assim: “**git add -all**” :

Texto

Descrição gerada automaticamente

* Para verificar se o git está acompanhando seu arquivo, dê o comando “**git status**”. Se estiver tudo certo, aparecerá esta mensagem, o que significa que seu arquivo criado está na camada Stage :

Texto

Descrição gerada automaticamente

* Depois que seus arquivos novos ou modificados estiverem na camada de Stage, você pode commita-los, ou seja, envia-los à ultima camada do fluxo, HEAD. Para isso use o comando “**git commit -m “tituloDoCommit”**. O parâmetro “-m” é passado para que você possa escrever uma descrição junto ao commit :

Texto

Descrição gerada automaticamente

* Agora a última etapa do fluxo é enviar o(s) arquivo(s) ao repositório remoto. Se você ainda não tem, crie um repositório remoto, pode ser no GitHub, GitLab ou qualquer outro de sua preferência.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

* Depois de criado, copie a URL HTTP do seu repositório, ela será necessária :



* Volte ao Git Bash escreva o comando: “**git remote add origin urlDoRepositorio**”. Com esse comando você está especificando o repositório ao qual você deseja manter o seu projeto. “Origin” é apenas um apelido padrão, mas você pode utilizar qualquer nome :

Texto

Descrição gerada automaticamente

* Depois de adicionado o repositório remoto, envie os arquivos utilizando o comando “**git push origin master**”.

Texto

Descrição gerada automaticamente

Master é o nome da branch padrão quando você cria um repositório, é onde ficará o seu código do produto final. Branch são ramificações do repositório, elas facilitam a organização do projeto, do time e garante segurança para o seu projeto final pois permitem que você trabalhe de forma isolada em diferentes funcionalidades e não tudo junto no produto final. Se você tiver outras branchs no seu repositório basta alterar o nome master para o nome da branch que você quer na hora do git push. Lembrando que antes de iniciar todo o fluxo, você ja deve estar na branch que deseja enviar suas alterações. Vou explicar melhor sobre Branches depois.

Processo finalizado. Após o git push seus novos arquivos ou novas alterações já estão no repositório, você pode ir até lá verificar.

**Branches**

Se você pretende desenvolver um projeto extenso ou então que abrange diferentes casos de uso, é recomendável que você separe seu projeto em branches. Você pode criar uma branch direto no seu repositório remoto, através da interface gráfica ou então através do Git Bash. Crie com o comando “**git checkout -b nomeDaBranch**”. O parâmetro “-b” significa git branch. Para navegar entre as branches utlize “**git checkout nomeDaBranch**”. Todo o processo descrito acima pode ser executado dentro das branches, conforme você for trabalhando nas diferentes funcionalidades do seu projeto.

Texto

Descrição gerada automaticamente

Este é um fluxo básico de funcionamento do Git porém suficiente para que você consiga trabalhar integrado com um repositório remoto. Existem diversas outras possibilidades escondidas aí neste fluxo. É interessante dar uma lida na documentação do próprio Git e caso seja de interesse seu, aprofundar os estudos.

# Estudando o GIT

## Compartilhar é se importar!

Compartilhe árvores com seus amigos usando export tree e import tree

Tem uma grande lição para compartilhar? Tente construir um nível com build level ou experimente o nível de um amigo com import level

Para ver todos os comandos, use show commands. Há algumas jóias como undo e reset

Por hora, vamos começar com o levels...

Comandos:

commit

branch

checkout

cherry-pick

reset

revert

rebase

merge

## Commits no Git

Um commit em um repositório git registra uma fotografia (snapshot) de todos os arquivos no seu diretório. É como um grande copy&paste, mas ainda melhor!

O Git tem por objetivo manter os commits tão leves quanto possível, de forma que ele não copia cegamente o diretório completo toda vez que você commita. Ele pode (quando possível) comprimir um commit como um conjunto de mudanças (ou um "delta") entre uma versão do seu repositório e a seguinte.

O Git também mantém um histórico de quando ocorreu cada commit. É por isso que a maioria dos commits tem ancestrais acima de si -- que indicamos usando setas na nossa visualização. Manter a história é ótimo para todos que trabalham no projeto!

Há muito para aprender, mas por enquanto pense nos commits como snapshots do seu projeto. Os commits são muito leves, e mudar de um para outro é extremamente rápido!

Criando uma Branch com o nome “newImage”

Comando: Git branch newImage

Mudando de uma Branch para outra

Comando: git checkout [nomeBranch]