**Utilização do Git Hub**

O Git Hub é um repositório de arquivos público, onde além de guardar os seus arquivos na nuvem, ele consegue controlar todas as versões / alterações geradas em cada um desses arquivos. Por essa facilidade de controle de versão de cada arquivo guardado, ele passou a ser utilizado pela comunidade de tecnologia para armazenamento de todos os arquivos fonte de um projeto de TI, pois além de armazenar, ele consegue “compartilhar” estes arquivos entre vários usuários e isso o tornou a ferramenta ideal de milhões de desenvolvedores de software em todo o mundo.

Para a sua utilização, assim como qualquer outro produto na nuvem, torna-se necessária a criação de uma conta pessoal, facilmente acessível por qualquer equipamento e em qualquer lugar onde você esteja logado.

1. Para criar uma conta no Git Hub entre no site [www.github.com](http://www.github.com) crie o seu login / senha e acesse todos os arquivos que você tenha enviado para ser gerido / controlado pelo Git Hub.
2. O Git Hub se organiza através de Repositórios que podem ser tratados como “Pastas ou Diretórios” de seu computador. Dentro de cada Repositório, você poderá ter um ou centenas de documentos a serem geridos dentro daquele repositório.
3. Por ser um produto de domínio público, todos os documentos armazenados com status “público” poderão e serão vistos / utilizados por centena de milhares de pessoas, o que torna o Git Hub, uma verdadeira comunidade de compartilhamento de conhecimento / códigos fonte desenvolvidos por milhões de pessoas em todo o mundo, por isso ele é o best-seller dos produtos utilizados pelos desenvolvedores de sistemas em todo o mundo.
4. Através do comando “Git Clone <url>”, você pode “baixar” todo o conteúdo de um repositório da nuvem, armazenado na <url> gerada pelo Git Hub, dentro de um diretório da sua própria máquina e ter acesso para editar, alterar o que for preciso para que este software atenda as necessidades que você precisa e em seguida, poderá “retornar” ou “gravar” novamente na nuvem, uma nova versão de seu documento, onde ficarão registradas todas as alterações realizadas, a data e hora das alterações gravadas e quem foi o responsável por essas alterações.

Obs: O Git Hub consegue identificar até, se 2 ou mais usuários diferentes, fizeram alterações “simultâneas” em uma mesma parte / sessão do documento, o que certamente irá gerar um “conflito” entre essas versões dentro do Git Hub. Nesse caso, para poder “armazenar” a versão correta do documento dentro da nuvem, o Git Hub irá “mostrar” a você, as sessões conflitantes do documento, de forma que você possa definir, qual versão deverá ficar “registrada” na nuvem, como sendo a versão atual e correta desse documento.

1. Para que tudo isso possa funcionar de forma correta, existe um padrão de comandos a serem utilizados, de forma que o Git Hub possa saber exatamente, o que gostaríamos de fazer, do tipo 🡺 me informe quais arquivos precisam ser atualizados na nuvem, faça as atualizações agora e envie os documentos para serem guardados agora na nuvem, e muito outros mais, que aprenderemos a seguir ...

**COMANDOS GIT HUB**

Git Clone <url> 🡺 “baixa” todos os arquivos do repositório <url> em sua máquina local.

Um diretório de nome <url> será criado em sua máquina.

Para executar outros comandos Git Hub para esse diretório, entre nele: CD \<url>

Git Add <arquivo> 🡺 Informe ao Git que um novo <arquivo> deverá ser adicionado ao Git Hub

Obs: Git Add <ponto> 🡺 Inclui todos os arquivos do <diretório>

Git Status 🡺 Mostra o status de todos os arquivos que tenham sido alterados ou adicionados

no seu diretório, após terem sido “baixados” através do comando “Git Clone”.

Obs: Arquivos “prontos” a serem gravados na nuvem, serão mostrados em “**verde**”

Git Commit -m “texto explicativo” 🡺 Avisa ao Git Hub que todos os arquivos do <diretório> estão

“gravados” e “prontos” para serem enviados para a nuvem. O “texto explicativo” deve

ser utilizado para informar o que está sendo alterado nesse novo Commit. Isso é muito

útil para que você e outras pessoas “entendam” o que está sendo mudado nessa nova

versão desses documentos.

Git Push origin <master> 🡺 Envia efetivamente todas as alterações dos documentos para a nuvem.

A opção <master> significa que esta alteração ocorre na Branch (ramo) principal de sua versão.

Qualquer outro nome utilizado no lugar do <master> significará uma nova Branch (ramificação)

do seu código. Essas inúmeras Branchs (ramificações) permitem alterações no código fonte,

sem alterar a sua versão original (são derivações de códigos para usos específicos), que poderão

ou não serem acopladas (mergeadas) ao código principal no futuro, se assim você desejar.

Obs: O Git Push irá solicitar o seu Login / Senha, para efetivar as alterações na nuvem.

Existe uma forma de “automatizar” essas alterações sem solicitação de login / senha,

mas precisa configurar a sua máquina como uma máquina “conhecida” no Git Hub,

através da chave SSH de sua máquina (settings SSHKeys – dentro do seu repositório

na nuvem e colocar a sua chave no diretório oculto .SSH da sua máquina).

Git Merge <branch> 🡺 Faz o merge da <branch> com a sua branch master

Obs: Possíveis conflitos serão mostrados na tela para serem resolvidos manualmente

Como desfazer um commit já efetuado na nuvem?

Entre no Git Hub e list todos os commits já realizados – todos possuem um Commit ID.

Git Checkout <commitid> 🡺 seu arquivo retornará para a versão anterior existente na nuvem

Git Checkout <master> 🡺 seu arquivo retornará para a última versão válida de sua Branch Master

Git Checkout <ponto> 🡺 ignora todas as modificações realizadas em sua máquina local

GitK / GitG / GitX 🡺 interface gráfica do Git, mostra a linha do tempo e todas as suas Branches

Git Gui 🡺 Framework Git para ajustar conflitos na versão dos seus arquivos

Git Reset 🡺 Limpa a timeline das branchs

Git Push -force 🡺 apaga tudo na nuvem

Git rm <arquivo> 🡺 apaga um arquivo

Git commit -am 🡺 adiciona e apaga arquivos removidos

Git log 🡺 mostra o log do git

Git Blame 🡺 mostra tudo o que foi feito no git

**Sumário no dia a dia do Git Hub**

Git Add <ponto>

Git commit -m “texto explicativo”

Git Status

Git Push

Git Status