Guia de Uso do GitHub/Git

Por Bernardo Netto

O que é?

O GitHub é uma plataforma hospedeira de códigos, a qual tem a função de ajudar as equipes de programadores a se organizar, compartilhando os seus códigos uns com os outros de forma mais usual. Nessa plataforma qual quer um pode usar para colocar os seus códigos e pedir ajuda, o que da a ela uma cara de rede social se precisar.

Essa plataforma é muito bem vista no mundo dos programadores a ponto de ser usada como referencia de biografia pessoal e fonte de base de dados do programador para ingressar em seus trabalhos.

Guia de uso:

Primeiro você deve instalar o Git ou alguma extensão que linca a sua API com o GitHub. Após isso o usuário já pode criar os seus projetos, criando um repositório para fazer o upload dos arquivos para dentro do projeto no GitHub

Funções básicas:

O Git possuí linhas de comandos as quais fazem a plataforma funcionar, após a criação da pasta em seu computador, os comandos são:

Configuração:

Setar User:

git config --global user.name "SL4K3R"

Setar E-mail:

git config --global user.email bernardomattos…@gmail.com

Setar Editor:

git config --global core.editor vim

Setar ferramenta de merge:

git config --global merge.tool vimdiff

Setar arquivos a serem ignorados:

git config --global core.excludesfile ~/.gitignore

Listar configurações:

git config –list

Repositório Local:

- Criar repositório local

git init

- Verificar estado dos arquivos/diretórios

git status

- Adicionar arquivo/diretório (staged area)

Adicionar um arquivo em especifico:

git add meu_arquivo.txt

Adicionar um diretório em especifico:

Git add meu_diretorio

Adicionar todos os arquivos/diretório:

git add .

Adicionar um arquivo que esta listado no .gitignore (geral ou do repositório)

git add -f arquivo_no_gitignore.txt

->Comitar arquivo/diretório

Comitar um arquivo:

git commit meu_arquivo.txt

Comitar vários arquivos:

git commit meu_arquivo.txt meu_outro_arquivo.txt

Comitar informando mensagem:

git commit meuarquivo.txt -m "minha mensagem de commit"

->Remover arquivo/diretório

Remover arquivo:

git rm meu_arquivo.txt

Remover diretório:

git rm -r diretório

Repositório remoto:

Exibir os repositórios remotos:

git remote

git remote -v

Vincular repositório local com um repositório remoto:

git remote add origin git@github.com:leocomelli/curso-git.git

Exibir informações dos repositórios remotos:

git remote show origin

Renomear um repositório remoto:

git remote rename origin curso-git

Desvincular um repositório remoto

git remote rm curso-git

- >Enviar arquivos/diretórios para o repositório remoto

O primeiro push de um repositório deve conter o nome do repositório remoto e o branch:

git push -u origin master

Os demais pushes não precisam dessa informação:

git push

->Atualizar repositório local de acordo com o repositório remoto

Atualizar os arquivos no branch atual:

git pull

Buscar as alterações, mas não aplica-las no branch atual:

git fetch

Clonar um repositório remoto já existente:

git clone git@github.com:leocomelli/curso-git.git

Tags:

Criando uma tag leve:

git tag vs-1.1

Criando uma tag anotada:

git tag -a vs-1.1 -m "Minha versão 1.1"

Criando uma tag assinada:

Para criar uma tag assinada é necessário uma chave privada (GNU Privacy Guard - GPG).

git tag -s vs-1.1 -m "Minha tag assinada 1.1"

Criando tag a partir de um commit (hash):

git tag -a vs-1.2 9fceb02

Criando tags no repositório remoto:

git push origin vs-1.2

Criando todas as tags locais no repositório remoto:

git push origin –tags

Branches:

O master é o branch principal do GIT.

O HEAD é um ponteiro especial que indica qual é o branch atual. Por padrão, o HEAD aponta para o branch principal, o master.

Criando um novo branch:

git branch bug-123

Trocando para um branch existente:

git checkout bug-123

Neste caso, o ponteiro principal HEAD esta apontando para o branch chamado bug-123.

Criar um novo branch e trocar:

git checkout -b bug-456

Voltar para o branch principal (master):

git checkout master

Resolver merge entre os branches:

git merge bug-123

Para realizar o merge, é necessário estar no branch que deverá receber as alterações. O merge pode automático ou manual. O merge automático será feito em arquivos textos que não sofreram alterações nas mesmas linhas, já o merge manual será feito em arquivos textos que sofreram alterações nas mesmas linhas.

A mensagem indicando um merge manual será:

Automerging meu_arquivo.txt

CONFLICT (content): Merge conflict in meu_arquivo.txt

Automatic merge failed; fix conflicts and then commit the result.

Apagando um branch:

git branch -d bug-123

Listar branches:

git branch

Listar branches com informações dos últimos commits:

git branch -v

Listar branches que já foram fundidos (merged) com o master:

git branch --merged

Listar branches que não foram fundidos (merged) com o master:

git branch --no-merged

Criando branches no repositório remoto:

Criando um branch remoto com o mesmo nome

git push origin bug-123

Criando um branch remoto com nome diferente:

git push origin bug-123:new-branch

Baixar um branch remoto para edição:

git checkout -b bug-123 origin/bug-123

Apagar branch remote:

git push origin:bug-123