Manual Git/GitHub HigFlow

HigFlow e HigTree

18 de novembro de 2022

1 Configurações iniciais

Para acessar o tutorial completo com as explicações mais detalhadas acesse [2]. As primeiras/recomendadas configurações para uso do *Github*, deve-se (após instalado o *git*) abrir um terminal e digitar:

- **git config --global user.name "Your Name"**
- **git config --global user.email your_email_github@example.com**

Para acessar o tutorial completo acesse [1]. Para conseguir fazer alterações no GitHub do grupo deve-se registrar uma chave *ssh* da máquina que você usa.

Para criar uma nova chave *ssh* da sua máquina, digite no terminal:

- **ssh-keygen -t ed25519 -C "your_email_github@example.com"**
- **eval "$(ssh-agent -s)"**
- **ssh-add ~/.ssh/id_ed25519**

Ainda no terminal, para visualizar a chave pública recém criada, digite:

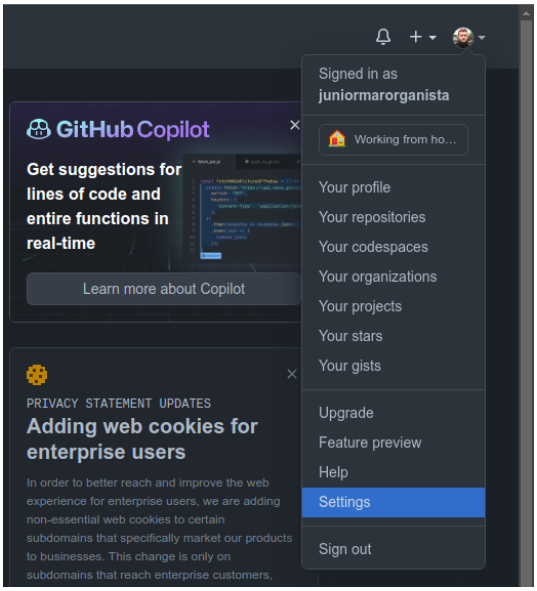
- **cat ~/.ssh/id_ed25519.pub**

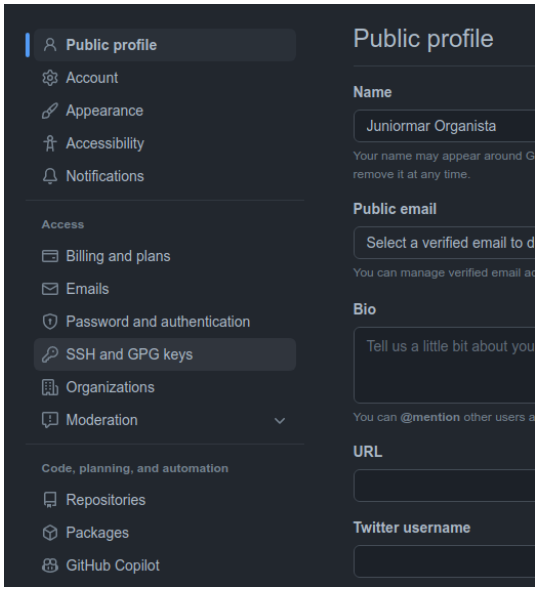
Este comando deve imprimir em tela algo parecido com:

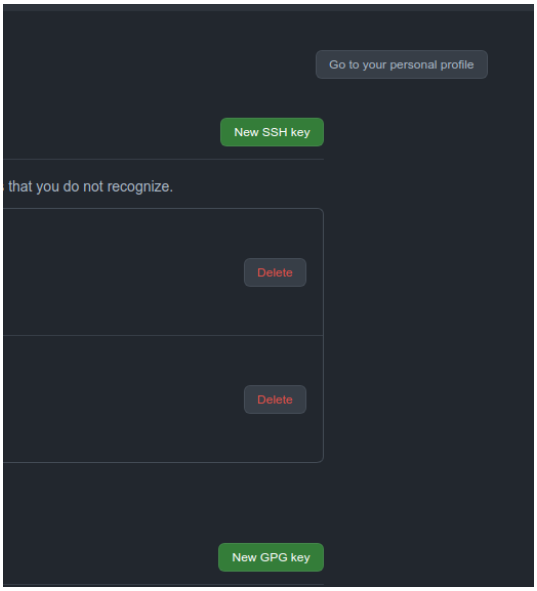
ssh-ed25519 LetrasAleat0riasENumer0s7amb3m your_email_github@example.com

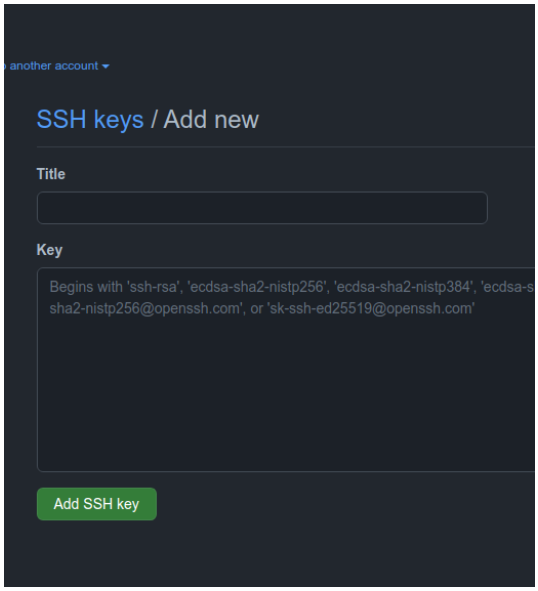
Copiei esta mensagem do seu terminal (essa é a parte publica da chave criada)! Posteriormente entre no seu GitHub no site e siga as seguintes instruções:

- **Acesse: Settings → SSH and GPG keys → New SSH key**
- No campo **Title**, utilize um nome para identificar a máquina que está utilizando e no campo **Key** cole a mensagem gerada no terminal ao digitar o comando **cat ~/.ssh/id_ed25519.pub**. Abaixo seguem algumas imagens dos passos acima:









Essas configurações servem para a maquina que foi criado a chave em questão. Caso utilize o projeto em mais de uma máquina, será necessário repetir esse processo na outra máquina. Na sua conta do GitHub terá todas as máquina que sua conta está associada. Existe ainda a possibilidade da máquina não suportar a tecnologia associada a criação da chave *ssh* com o algoritmo ***Ed25519***. Caso isso ocorra, será necessário criar uma chave do tipo *rsa*, para será necessário trocar a criação da chave de

- **ssh-keygen -t ed25519 -C "your_email_github@example.com"**

por

- **ssh-keygen -t rsa 4096 -C "your_email_github@example.com"**

2 Projeto HigFlow no GitHub - Primeiro passos

O primeiro passo para se ter acesso ao projeto HigFlow no GitHub é solicitar a permissão ao administrador do projeto, Prof. Dr. Antonio Castelo Filho, através do email: **castelo@icmc.usp.br**. Após receber o convite para o projeto e aceita-lo, siga os passos abaixo pra ter uma versão na maquina que deseja trabalhar (que já tenha realizado as instruções da seção 1). Primeiramente abra um terminal no diretório que deseja baixar o projeto e digite o comando abaixo:

- **git clone git@github.com:antoniocastelofilho/HigFlow.git**

Para iniciar suas alterações no projeto é necessário que se crie um ***Branch***(ramo/versão) com algum titulo que identifique o usuário. Posteriormente quando possuir uma versão amplamente testada e estável será realizado um merge no código principal.

3 Comandos básicos do Git para controle de versão

Alguns comandos de uso cotidiano (para maiores informações consulte [3]):

- **git init** := Cria o controle de versões git na pasta.
- **git status** := Exibe o status dos arquivos controlados e não controlados.
- **git add -A** := Adiciona todos os arquivos da pasta para controle.
- **git add poisson.m** := Adiciona o arquivo 'poisson.m' pasta para controlar.
- **git add -u** := Adiciona somente os arquivos já adicionados e que foram modificados.
- **git commit -m "Comentario da versao"** := Faz os arquivos adicionados serem controlados pelo git e adiciona o comentário 'Comentario da versao'.
- **git checkout -b add-var-in-main** := Cria um ***Branch***(ramo) com o nome 'add-var-in-main', que na pratica pode ser usado para voltar um determinado arquivo antes de algumas alterações.
- **git checkout master** := Muda para o Branch denominado 'master'.
- **git merge teste_merge -m "teste de domingo"** := Quando no Branch 'master' este comando unifica a versão contida no branch 'teste_merge' ao 'master' com o comentário 'teste de domingo'.
- **git log --oneline --decorate --all --graph** := Mostra um diagrama do estado de controle dos arquivos.
- **git config --global alias.tree "log --oneline --decorate --all --graph"** := Cria o comando 'tree' que executa o comando 'log --oneline --decorate --all --graph'.
- **git tree** := Executa o comando 'tree' criado.
- **sudo apt install gitk** := Instala um visualizador gráfico.
- **gitk** := Executa o visualizador gráfico.

- **git remote add origin https://github.com/your_user/project.git** := Usado para adicionar um 'origin' para o repositório se não existe nenhum.
- **git pull origin master** := Caso o 'origin' já esteja adicionado na pasta, este comando é usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local.
- **git push -u origin master** := É usado para enviar conteúdo do repositório local para um repositório remoto.
- **git clone https://github.com/your_user/project.git** := É utilizado para selecionar um repositório existente e criar um clone ou cópia do repositório alvo (no caso GitHub).

Referências

[1] https://docs.github.com/pt/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent.

[2] https://git-scm.com/book/en/v2/getting-started-first-time-git-setup.

[3] https://git-scm.com/doc.