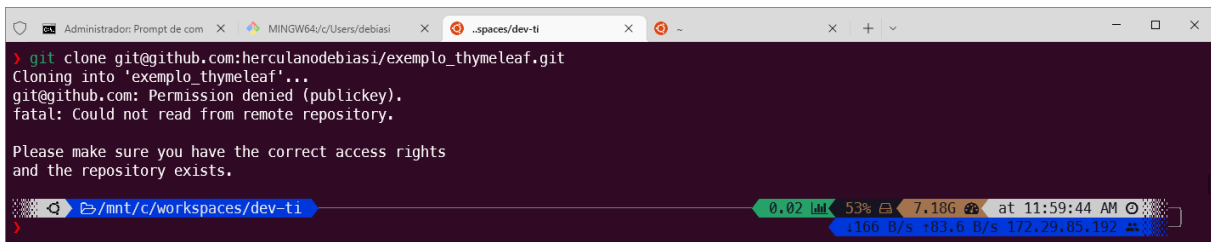


1 Git

1.1 Motivação

Erro ao tentar clonar um repositório do GitHub.

A screenshot of a Windows terminal window. The title bar shows several open tabs: 'Administrador: Prompt de com...', 'MINGW64/c/Users/debiasi', and '.spaces/dev-ti'. The terminal content shows a command prompt where the user has run 'git clone git@github.com:herculanodebiasi/emplo_thymeleaf.git'. The output shows the cloning process starting, but then failing with the error: 'git@github.com: Permission denied (publickey). fatal: Could not read from remote repository.' Below the error, a message says 'Please make sure you have the correct access rights and the repository exists.' The terminal's status bar at the bottom shows the current directory as 'D:/mnt/c/workspaces/dev-ti', memory usage at 0.02 GB, 53% disk usage, and a file size of 7.186 GB. The time is 11:59:44 AM.

1.2 Segurança

Ao baixar um repositório no GitHub, temos dois protocolos à nossa disposição: 1) HTTPS e 2) SSH. O repositório em SSH inicia seu endereço com “git@github.com”, já o HTTPS inicia com “https://github.com”. A primeira opção permite que trafegemos dados de maneira segura para nosso GitHub. Quando usamos esse protocolo, precisamos informar nosso usuário e senha do GitHub sempre que desejarmos adicionar novos arquivos ao nosso repositório remoto. Isso é muito seguro e funciona bem. Contudo, com o passar do tempo, esse processo pode se tornar um “problema”, pois o processo repetitivo de informar nossas credenciais pode se tornar cansativo.

Mas, quando usamos a segunda opção, o SSH, não precisamos informar nosso usuário e senha em cada acesso. Basta configurar nossas credenciais uma vez e depois sempre poderemos acessar nosso GitHub sem informá-las, pois o SSH é usado especialmente para realizar a conexão com servidores remotos.

O que é preciso fazer é criar uma chave SSH em nosso computador local e adicioná-la ao GitHub. Essa chave é como uma credencial. Para entendermos melhor, pense nessa ilustração: quando vamos a algum evento, talvez haja alguma catraca ou sistema de verificação na entrada do evento. Nós só podemos entrar no evento se possuímos uma credencial, geralmente um crachá com nossos dados. Podemos dizer que uma chave SSH é como esse “crachá”, que vai nos permitir acessar o GitHub remotamente.

O SSH é usado para transferência remota de arquivos, gerenciamento de rede e acesso remoto ao sistema operacional. O acrônimo SSH também é usado para descrever um conjunto de ferramentas usadas para interagir com o protocolo SSH. O SSH usa um par de chaves para iniciar um aperto de mãos seguro entre partes remotas. O par de chaves contém uma chave pública e privada. A nomenclatura privada versus pública pode ser confusa, pois ambas são chamadas de chaves. É mais útil pensar na chave pública como “fechadura” e na chave privada como a “chave”. Você concede o “bloqueio” público a partes remotas para criptografar ou “bloquear” os dados. Esses dados são abertos com a chave “privada” que você mantém em local seguro.

As chaves SSH são geradas por meio de algoritmo criptográfico de chave pública, sendo o mais comum o RSA ou o DSA. As chaves SSH são geradas por meio de fórmula matemática que leva 2 números primos e uma variável de semente aleatória para gerar a chave pública e privada. Essa é uma fórmula unidirecional que garante que a chave pública possa ser derivada da chave privada, mas a chave privada não pode ser derivada da chave pública.

As chaves SSH são criadas usando uma ferramenta de geração de chaves. O conjunto de ferramentas de linha de comando SSH inclui uma ferramenta de keygen. Os sistemas operacionais OSx e Linux têm aplicativos de terminal modernos e abrangentes que vêm com o pacote SSH instalado. O processo para criar chave SSH é o mesmo entre eles. Os ambientes Windows não têm um *shell* Unix padrão. Programas de *shell* externos vão precisar ser instalados para ter a experiência completa de keygen. A opção mais simples é utilizar o Git Bash.

Observação: o GitHub aprimorou a segurança removendo tipos de chaves mais antigos e não seguros em 15 de março de 2022.

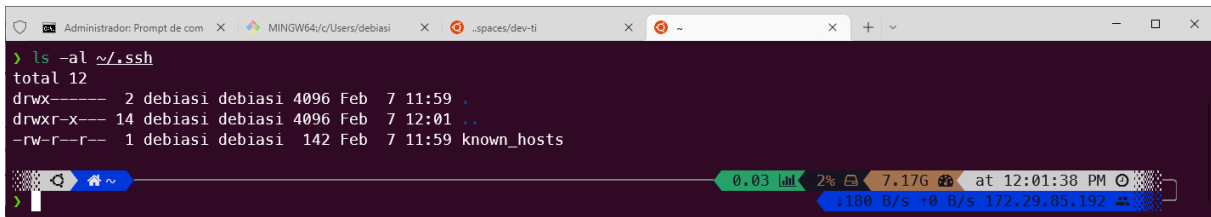
Desde essa data, não há mais suporte para as chaves DSA (`ssh-dss`). Não é possível adicionar novas chaves DSA à conta pessoal em GitHub.com.

As chaves RSA (`ssh-rsa`) com um `valid_after` antes de 2 de novembro de 2021 podem continuar usando qualquer algoritmo de assinatura. As chaves RSA geradas após essa data precisam usar um algoritmo de assinatura SHA-2. Talvez alguns clientes mais antigos precisem ser atualizados para usar as assinaturas SHA-2.

1.3 Geração das Chaves

Verificando através do terminal se as chaves já existem:

```
➤ ls -al ~/.ssh
```

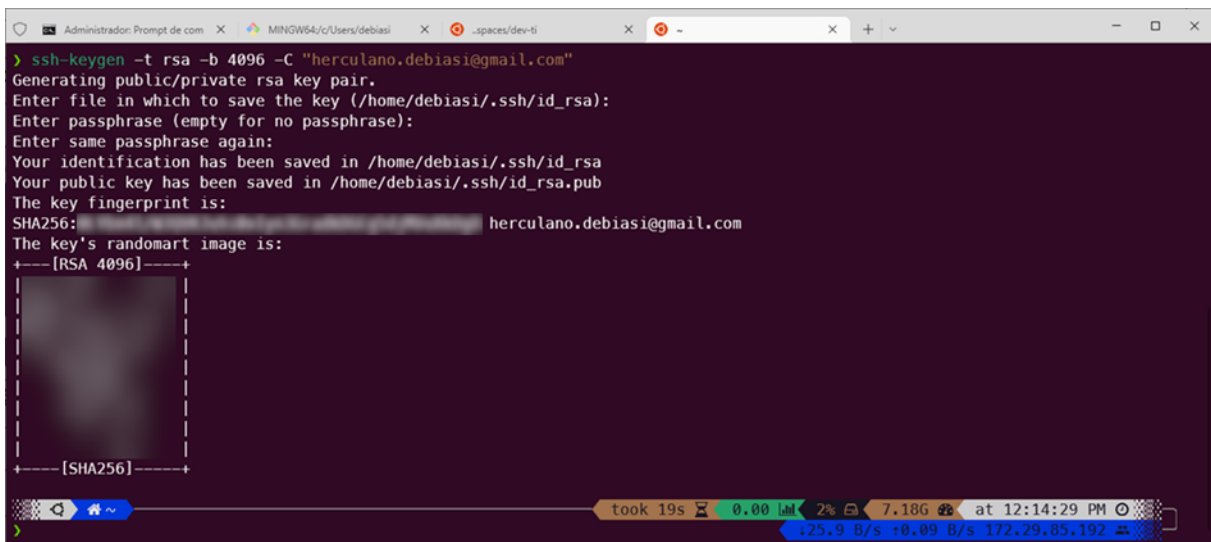


```
> ls -al ~/.ssh
total 12
drwx----- 2 debiasi debiasi 4096 Feb  7 11:59 .
drwxr-x--- 14 debiasi debiasi 4096 Feb  7 12:01 ..
-rw-r--r--  1 debiasi debiasi  142 Feb  7 11:59 known_hosts
```

Para gerar as chave execute o comando abaixo alterando para a sua conta de *e-mail* cadastrada no GitHub.

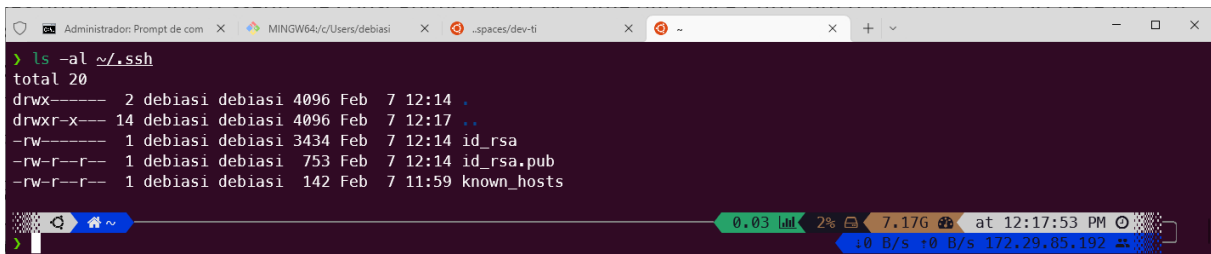
```
➤ ssh-keygen -t rsa -b 4096 -C "conta@gmail.com"
```

Resultado:



```
> ssh-keygen -t rsa -b 4096 -C "herculano.debiasi@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/debiasi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/debiasi/.ssh/id_rsa
Your public key has been saved in /home/debiasi/.ssh/id_rsa.pub
The key fingerprint is:
SHA256: herculano.debiasi@gmail.com
The key's randomart image is:
+--[RSA 4096]--+
|
|
|
|
|
|
|
|
|
|
+-----[SHA256]-----+
```

Chave criada com sucesso!



```
> ls -al ~/.ssh
total 20
drwx----- 2 debiasi debiasi 4096 Feb  7 12:14 .
drwxr-x--- 14 debiasi debiasi 4096 Feb  7 12:17 ..
-rw-----  1 debiasi debiasi 3434 Feb  7 12:14 id_rsa
-rw-r--r--  1 debiasi debiasi  753 Feb  7 12:14 id_rsa.pub
-rw-r--r--  1 debiasi debiasi  142 Feb  7 11:59 known_hosts
```


O ssh-agent é outro programa que faz parte do conjunto de ferramentas SSH. O ssh-agent é responsável por manter as chaves privadas. Pense nisso como um chaveiro. Além de manter chaves privadas, ele também intermedia solicitações para assinar solicitações SSH com as chaves privadas para que as chaves privadas nunca sejam transmitidas com insegurança.

Antes de adicionar a nova chave SSH ao ssh-agent, verifique se o ssh-agent está sendo executado:

```
➤ eval "$(ssh-agent -s)"
```

Após isso acrescente a chave (no macOS acrescente o parâmetro -K)

```
➤ ssh-add ~/.ssh/id_rsa
```



```
> eval "$(ssh-agent -s)"
Agent pid 469
> ssh-add ~/.ssh/id_rsa
Enter passphrase for /home/debiasi/.ssh/id_rsa:
Identity added: /home/debiasi/.ssh/id_rsa (herculano.debiasi@gmail.com)
```

A nova chave SSH agora está registrada e pronta para uso!

1.4 Adicionando a Chave SSH ao GitHub

Primeiramente copie a chave SSH criada há pouco na área de transferência.

No Windows, execute o seguinte:

```
> clip < ~/.ssh/id_rsa.pub
```

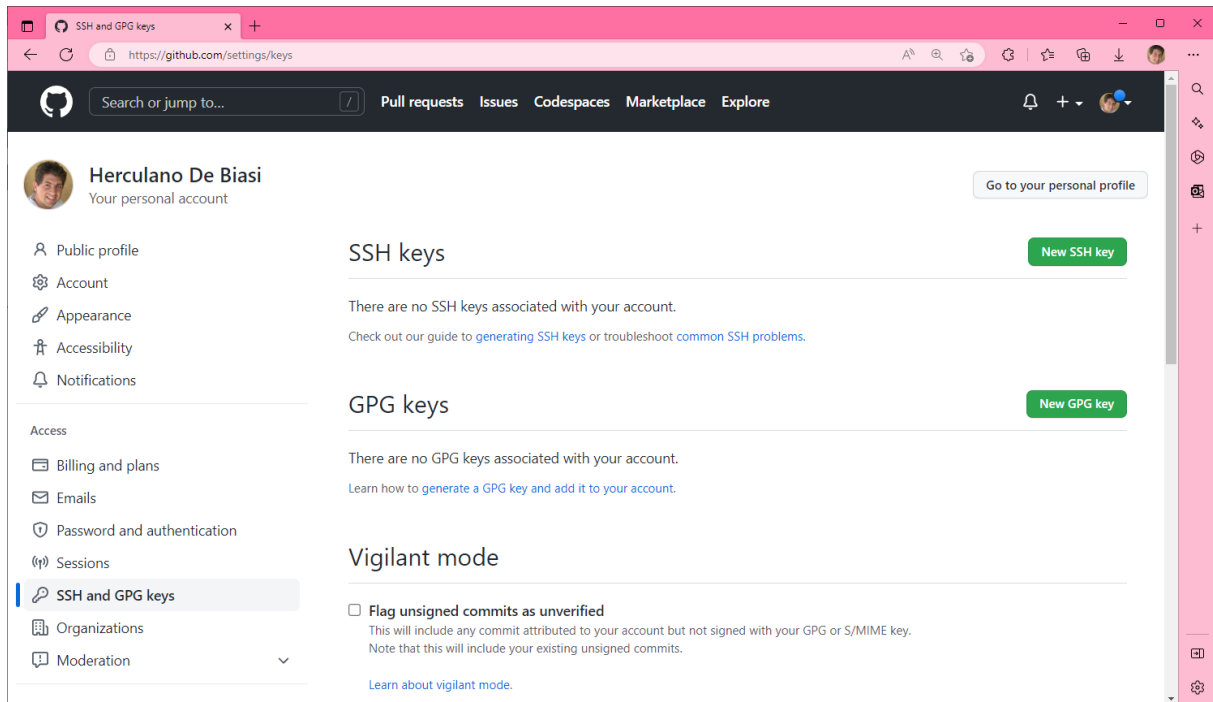
Se estiver usando o macOS, execute o comando abaixo:

```
> pbcopy < ~/.ssh/id_rsa.pub
```

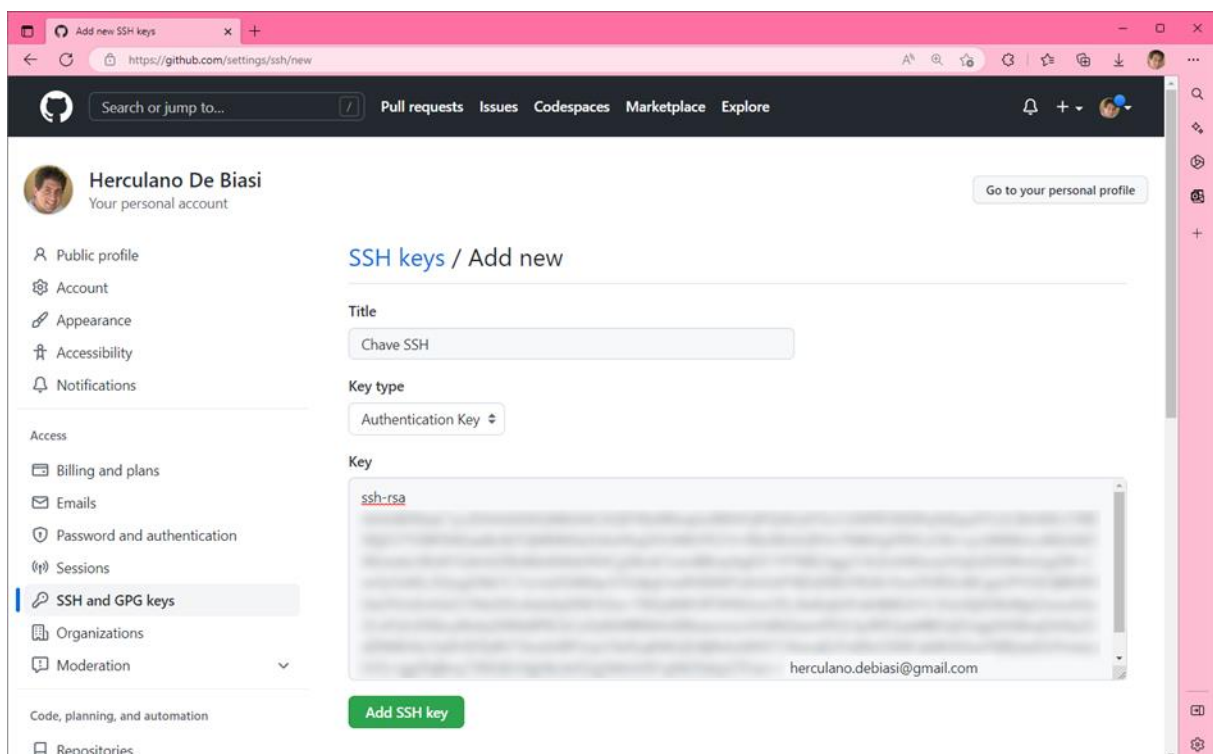
Mas, se seu computador for Linux, será necessário instalar antes o programa xclip para realizar o processo de cópia do arquivo. Execute este comando:

```
> sudo apt-get install xclip  
> xclip -sel clip < ~/.ssh/id_rsa.pub
```

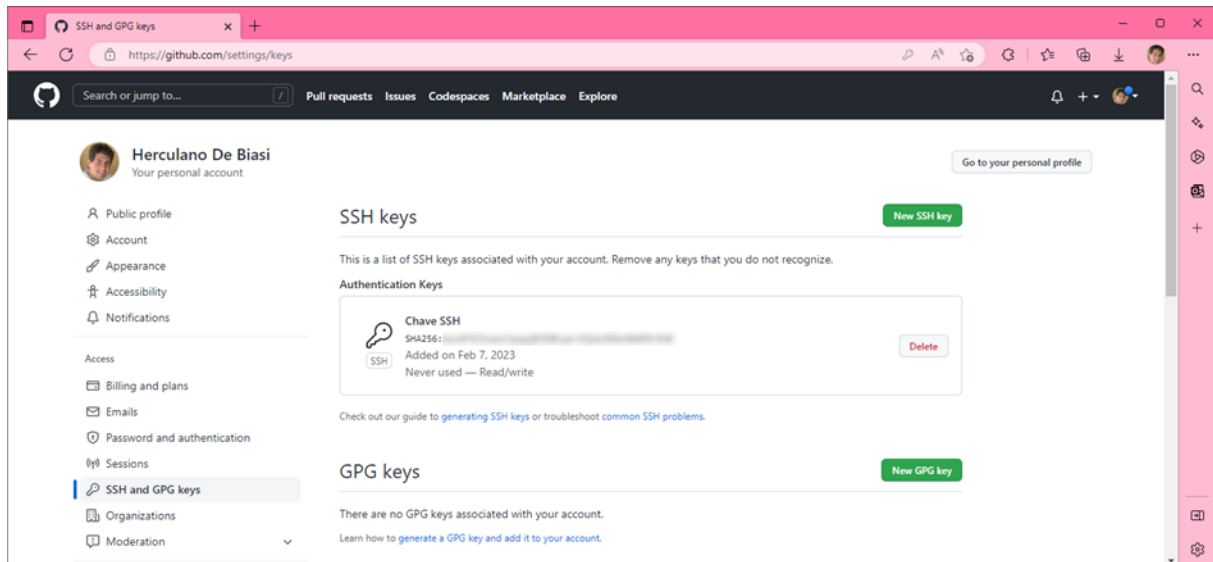
Abra esta [página](#) no GitHub e clique em *New SSH key*.



Agora que estamos com o conteúdo do arquivo copiado, vamos colá-lo no GitHub e clicar em *Add SSH key*:



Chave adicionada:



Repositório clonado com sucesso:

```
git clone git@github.com:herculanodebiasi/exemplo_thymeleaf.git
Cloning into 'exemplo_thymeleaf'...
Enter passphrase for key '/home/debiasi/.ssh/id_rsa':
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 43 (delta 3), reused 43 (delta 3), pack-reused 0
Receiving objects: 100% (43/43), 61.45 KiB | 469.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.
```

2 Git com Linha de Comando