

## STREAM SERIES

# devnator

Acesso remoto SSH com senha ou chaves de criptografia

SERGIO CABRAL



[sergiocabral.com](http://sergiocabral.com)

# Índice

1. Conexão com senha .....	2
2. Conexão com chaves de criptografia.....	3
3. Alternativas para clientes SSH .....	5
4. Demonstração em vídeo .....	6

## **Objetivo prático desta demonstração**

Estabelecer uma conexão via SSH, ou usando senha, ou usando um par de chaves de criptografia.

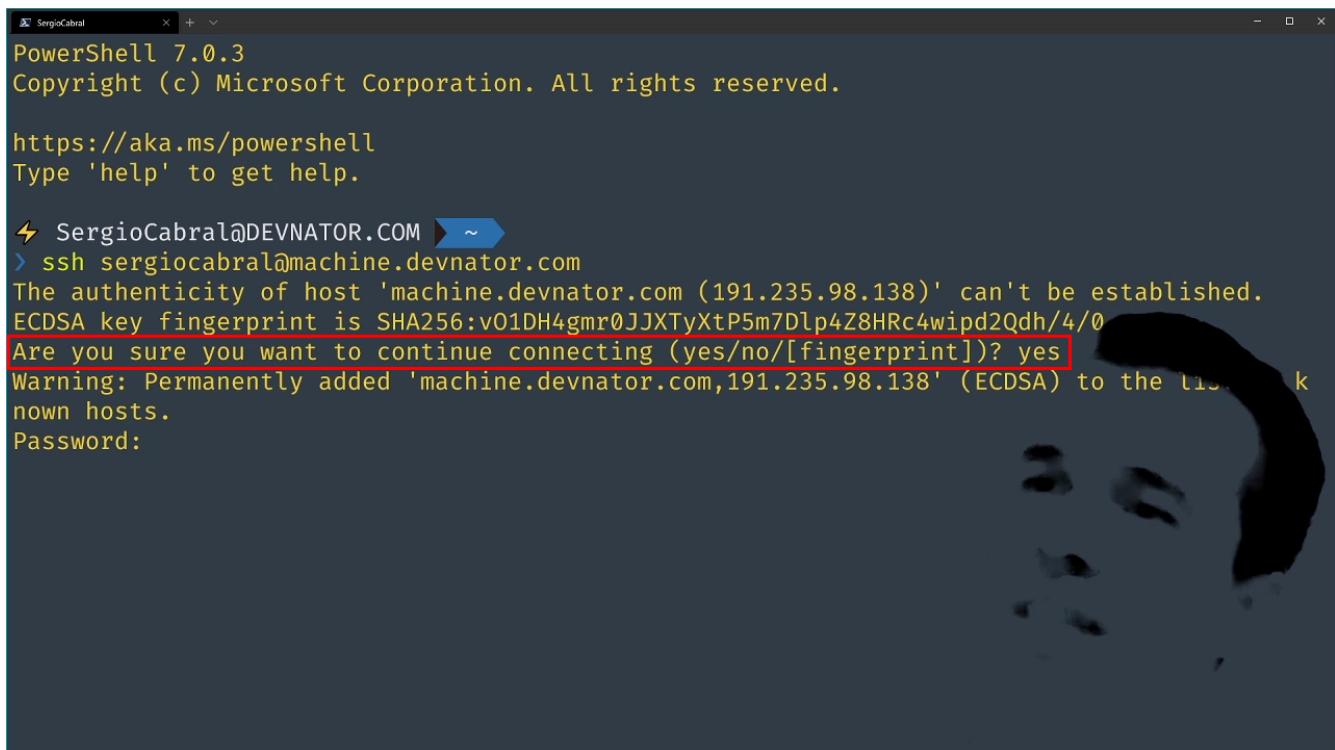
# 1. Conexão com senha

Para se conectar a um servidor com o cliente nativo do Windows 10, digite:

```
ssh nome-do-usuario@endereço-do-servidor
```

Existem outros clientes SSH famosos como indicado na seção [Alternativas para clientes SSH](#). Mas o cliente nativo do Windows 10 provavelmente já será o suficiente para o que você precisa fazer.

Quando é o primeiro acesso você precisa informar que confia na identidade do computador remoto respondendo **yes**, como indicado na *Figura 1*.



A screenshot of a PowerShell window titled "SergioCabral". The window shows the following text:

```
PowerShell 7.0.3
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/powershell
Type 'help' to get help.

⚡ SergioCabral@DEVNATOR.COM ~
> ssh sergiocabral@máquina.devnator.com
The authenticity of host 'máquina.devnator.com (191.235.98.138)' can't be established.
ECDSA key fingerprint is SHA256:v01DH4gmr0JJXTyXtP5m7Dlp4Z8HRc4wipd2Qdh/4/0
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'máquina.devnator.com,191.235.98.138' (ECDSA) to the list of known hosts.
Password:
```

The line "Are you sure you want to continue connecting (yes/no/[fingerprint])? yes" is highlighted with a red box.

Figura 1. Confirmação no primeiro acesso ao computador remoto

Ao fazer isso a chave de criptografia pública do computador remoto é armazenada no arquivo `~/.ssh/known_hosts` do computador local.

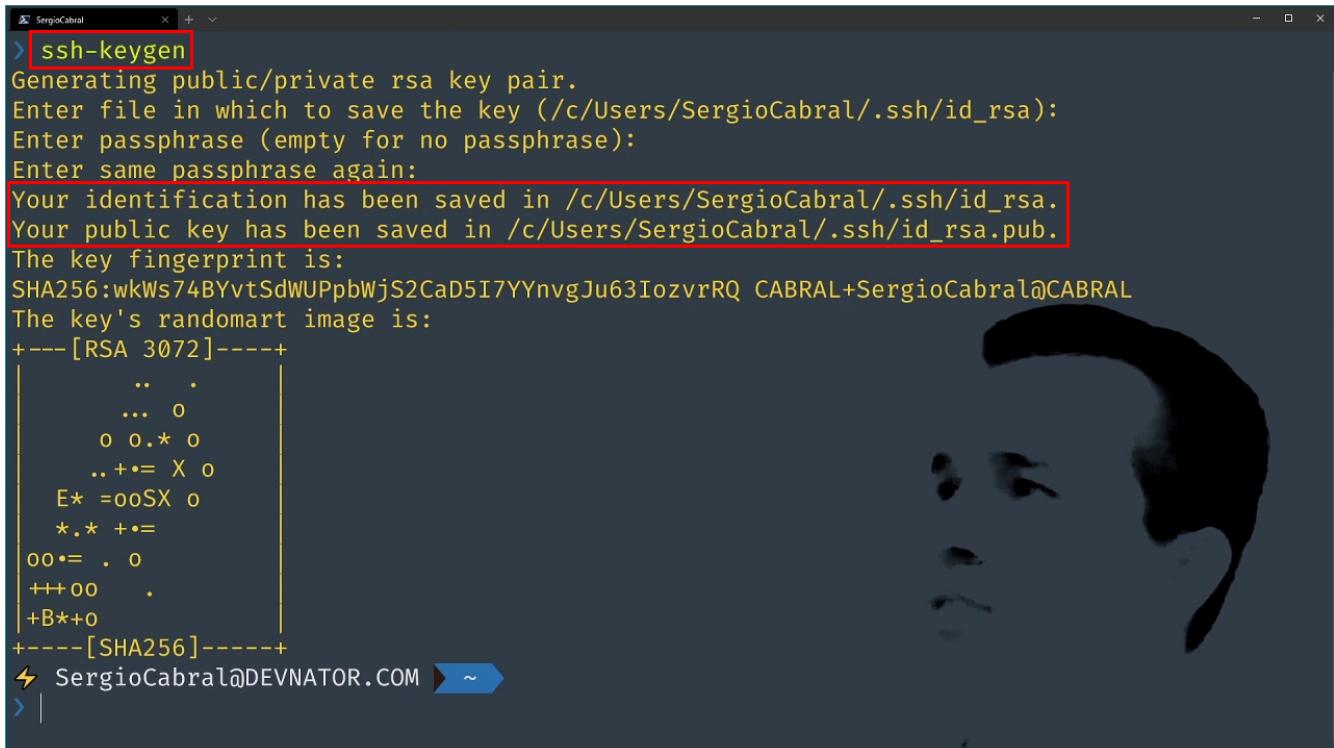


Se este arquivo for apagado ou tiver a linha que contém a chave pública do computador remoto removida, então a mensagem de confirmação voltará a ser exibida.

Na sequência você informa sua senha, e pronto. Conectado!

## 2. Conexão com chaves de criptografia

Outra forma de autenticação é fazer com que o computador de destino conheça o computador de origem. Neste caso criamos um par de chaves de criptografia no computador de origem com o comando `ssh-keygen`. Execute e pressionar `Enter` até concluir, como mostrado na *Figura 2*.



```
SergioCabral
> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/SergioCabral/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/SergioCabral/.ssh/id_rsa.
Your public key has been saved in /c/Users/SergioCabral/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:wkWs74BYvtSdWUPpbWjS2CaD5I7YYnvgJu63IozvrRQ CABRAL+SergioCabral@CABRAL
The key's randomart image is:
+---[RSA 3072]---+
| .. .
| ... o
| o o.* o
| ..+= X o
| E*=ooSX o
| *.* +=
| oo*= . o
| +++oo . .
| +B**o
+---[SHA256]---+
⚡ SergioCabral@DEVNATOR.COM ➔ ~
```

Figura 2. Execução padrão do comando "ssh-keygen"



Se você executar novamente o comando `ssh-keygen`, por padrão (ao pressionar `Enter` até o final) ele não sobreescreve as chaves geradas anteriormente. Mas se você especificar que deseja fazer isso vai perder qualquer acesso a serviços que dependiam daquelas chaves. É uma operação irreversível.

Como também indicado na *Figura 2*, um par de arquivos é gerados. Nos interessamos para esta demonstração no arquivo público `id_rsa.pub`. O arquivo privado `id_rsa` deve ser mantido seguro e nunca compartilhado.

```

*.* +:=
00*= . o
++oo .
+B**o
-----[SHA256]-----
⚡ SergioCabral@DEVNATOR.COM ~
> type /Users/SergioCabral/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQCesxW56XDbbmvxkIV5wtRLVuZgiKchtaJ/Zj0/+RPrsY5DHZBhn
Hzi1egWvxl8CP76shM40XXLFv19Uz5qbck0hoJvE06mG0Dlz1au3pS4nfbfDMrIUvYIDR5mgt+17fna0zLItVK6K
1BAYpVK7007HHXRToEbMsNgwpGA46Bb8LdlWNmyGEY00wY2FUCpJ0wrNzGzjOPMKI8aMT...
t92yumz7mw+hZy2Fn4top2MfTRSrHJvqLTqd4NcLpNdAXqGwc9/YwBKID9U7iWXL+CA
0kR4rggfst8eEzOkHLZ/hQz/6exY8zjeDaEt01dvHnbygnHo7DHQsBb5Xfc6DEN0IFMhLMntHwWqIS...
DQaf+RqHELmWDEIhU56MShgpEI2j8mq5cHlHQsOUxAo03CVXZtQvNTYZ3MVpChl5PQ7ZXUggQqYi6dgce...
vgrO6TEopihGN7mV0= CABRAL+SergioCabral@CABRAL
⚡ SergioCabral@DEVNATOR.COM ~
> ssh sergiocabral@machine.devnator.com
Password:
Linux devnator 4.19.0-10-cloud-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64

Last login: Tue Sep 29 23:54:17 2020 from 177.79.120.44
sergiocabral@devnator:~$ mkdir .ssh
sergiocabral@devnator:~$ cd .ssh
sergiocabral@devnator:~/ssh$ echo "" >> authorized_keys

```

Figura 3. Conteúdo do arquivo "id\_rsa.pub"

Então, enviamos o conteúdo do arquivo `id_rsa.pub`, como exemplificado na *Figura 3*, para o computador de destino, o servidor. Esse conteúdo deve ser adicionado ao arquivo `~/.ssh/authorized_keys`. Caso não exista deve ser criado.

Como mostrado na *Figura 3*, o conteúdo da chave de criptografia pública é um texto curto que você pode copiar usando o mouse e a combinação de comandos `type` ou `cat` para exibir e `echo` para escrever.

Mas caso esteja sem mouse, talvez prefira copiar o arquivo diretamente com o comando `scp` e, então, fazer a adição da chave no arquivo `authorized_keys`:



Comando no computador de origem:

- `scp ~/.ssh/id_rsa.pub nome-do-usuario@endereço-do-servidor:~/.ssh`

Comando no computador de destino:

- `cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`

A partir de agora tanto o seu computador conhece o servidor, como o servidor conhece o seu computador. Cada computador tem a chave pública do outro.

Faça uma nova tentativa de conexão e pronto. Conectado com o uso de chaves sem precisar informar dados de autenticação.

### 3. Alternativas para clientes SSH

Nome	Licença	Download
PuTTY	free; open-source	<a href="http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html">http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html</a>
SuperPutty	free; open-source; baseado no PuTTY	<a href="https://github.com/jimradford/superputty">https://github.com/jimradford/superputty</a>
PuTTY Tray	free; open-source; baseado no PuTTY	<a href="https://puttytray.goeswhere.com/">https://puttytray.goeswhere.com/</a>
KiTTY	free; open-source; baseado no PuTTY	<a href="http://www.9bis.net/kitty/">http://www.9bis.net/kitty/</a>
MobaXterm	free; versão Pro disponível para compra	<a href="http://mobaxterm.mobatek.net/">http://mobaxterm.mobatek.net/</a>
SmarTTY	free	<a href="http://smartty.sysprogs.com/">http://smartty.sysprogs.com/</a>
Dameware SSH client	free; versões disponíveis para compra	<a href="http://www.dameware.com/free-ssh-client-for-windows.aspx">http://www.dameware.com/free-ssh-client-for-windows.aspx</a>
mRemoteNG	free; open-source	<a href="http://www.mremoteng.org/">http://www.mremoteng.org/</a>
Terminals	free; open-source	<a href="https://terminals.codeplex.com/">https://terminals.codeplex.com/</a>
Secure Shell App	free; Chrome Addon	<a href="https://chrome.google.com/webstore/detail/pnhechapfaindjhompbnflcldabghjo">https://chrome.google.com/webstore/detail/pnhechapfaindjhompbnflcldabghjo</a>

## 4. Demonstração em vídeo

<https://youtu.be/64WqbNNJ0n4>

Hasta la vista.