

ENTRE PORTAS E TÚNEIS

ALÉM DOS HORIZONTES DO SSH LOGIN



Juliano Lopes

ENTRE PORTAS E TÚNEIS

ALÉM DOS HORIZONTES DO SSH LOGIN

A digital tunnel with binary code on the floor and open doors on the sides.

Juliano Lopes

Juliano Lopes

©2025

Obra licenciada sob uma Licença Creative Commons Atribuição 4.0 Internacional.

Sumário

Capítulo 1 - Conectando mundos: O problema que o SSH resolve

- Breve contexto histórico
- Um pouco sobre IPs e conexões
- A transição para o IPv6
- E onde o SSH entra nisso?

Capítulo 2 - Muito além do login: o SSH no dia a dia

- O login básico com SSH
- Login direto com endereço IP
- A autenticação por chaves
- Boas práticas de segurança

Capítulo 3 - Entre portas e túneis: o poder do port forwarding

- Abrindo caminhos seguros dentro da rede
- Port Forwarding Local
- Port Forwarding Remoto
- Port Forwarding Dinâmico

01

Conectando mundos: o problema que o SSH resolve

Antes do SSH, a internet era um mundo de portas abertas sem fechadura.

Conectando mundos: O problema que o SSH resolve

Breve contexto histórico

Nos primórdios da internet, a comunicação entre computadores era feita de maneira aberta e desprotegida. Protocolos como Telnet e rlogin permitiam que usuários acessassem máquinas remotas, mas havia um grande problema:

- Tudo era transmitido em texto puro, incluindo senhas e dados sensíveis.

Isso significava que qualquer pessoa com um pouco de conhecimento técnico poderia interceptar o tráfego e ler as informações facilmente. Imagine enviar sua senha escrita em um papel sem envelope, passando de mão em mão até chegar ao destino. Era exatamente isso que acontecia.

Foi nesse cenário que, em 1995, o pesquisador finlandês Tatu Ylönen criou o SSH (Secure Shell). A proposta era simples, mas revolucionária: criar um túnel criptografado para que ninguém pudesse espionar ou alterar os dados durante a comunicação.

Um pouco sobre IPs e conexões

Antes de entrar no SSH, vale uma explicação rápida sobre como os computadores “se encontram” na rede. Pense em um endereço IP como o endereço da sua casa digital. Quando você acessa outro computador, na prática está dizendo: “quero visitar a casa com este endereço”.

O SSH funciona como um carro blindado que leva sua mensagem de um endereço ao outro, sem deixar que ninguém a espione no caminho.

Para que dois computadores conversem na rede, eles precisam de um endereço, isso é o que chamamos de endereço IP (Internet Protocol). Pensando nisso, é importante entendermos a estrutura de um IP.

Estrutura básica do IPv4:

O formato mais conhecido é o IPv4, criado lá em 1981. Ele é formado por quatro blocos de números separados por pontos, onde cada bloco pode ir de 0 a 255.

Exemplo:



```
192.168.1.10
```

Aqui temos quatro blocos: `192` . `168` . `0` . `1`

Alguns endereços são reservados para uso interno, como a faixa 192.168.x.x, que você provavelmente já viu no Wi-Fi de casa. Outros endereços são públicos, e funcionam como o endereço real da sua rede na internet.

A transição para o IPv6

Com o crescimento absurdo da internet, o IPv4 ficou pequeno.

Ele suporta cerca de 4 bilhões de endereços únicos. Parece muito, mas não é quando pensamos em bilhões de celulares, computadores, IoTs, servidores, etc.

Para resolver isso, surgiu o IPv6. Ele usa 8 blocos de números e letras hexadecimais (0 a 9 + A a F). É gigantesco em capacidade, suportando 340 undecilhões de endereços (um número com 36 zeros).

Exemplo:



```
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

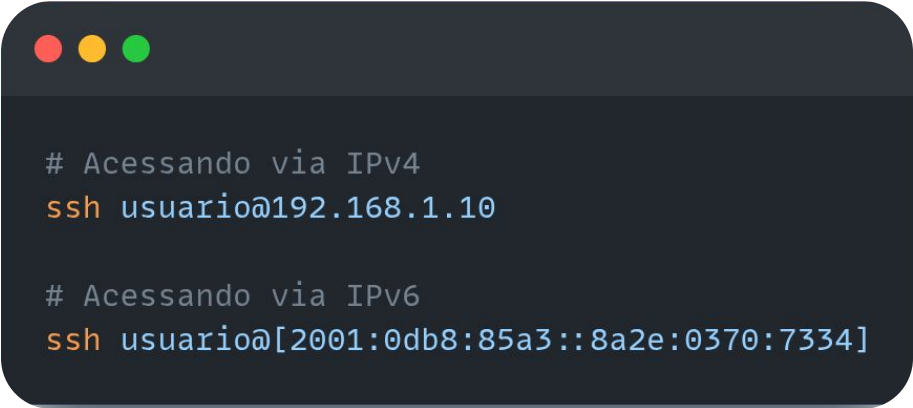
A transição ainda está em andamento:

Muitos lugares usam IPv4 com técnicas de reaproveitamento (como NAT). Outros já estão 100% prontos para IPv6.

E onde o SSH entra nisso?

O SSH pode funcionar tanto em IPv4 quanto em IPv6. Ou seja, não importa se você está acessando um servidor com endereço tradicional (IPv4) ou moderno (IPv6), o SSH consegue criar o mesmo túnel seguro para sua conexão.

Exemplos:



```
# Acessando via IPv4
ssh usuario@192.168.1.10

# Acessando via IPv6
ssh usuario@[2001:0db8:85a3::8a2e:0370:7334]
```

Assim, você mostra que um IP é apenas o ponto de partida da conversa digital. O SSH é quem vai garantir que essa conversa seja protegida e privada, não importa se você está em IPv4 ou IPv6.

02

Muito além do login: o SSH no dia a dia

O SSH não é apenas uma chave para entrar, é também um escudo que protege.

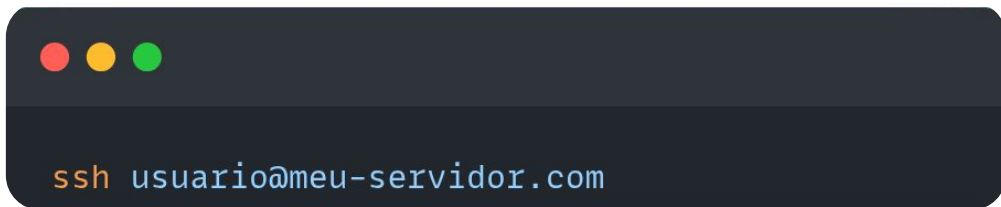
Muito Além do Login: O SSH no Dia a Dia

Quando pensamos em SSH, a primeira imagem que vem à cabeça é a de um terminal preto, onde digitamos o usuário, senha e acessamos um servidor distante. Esse é, de fato, o uso mais comum: o login remoto seguro. Mas o SSH vai muito além disso.

Com ele, é possível administrar servidores em qualquer lugar do mundo, transferir arquivos, criar túneis de comunicação e até simular uma rede privada segura. Ainda assim, tudo começa com o básico: o ato de se conectar.

O login básico com SSH

A forma mais simples de uso do SSH é se conectar informando usuário e endereço do servidor. O comando é direto, quase auto-explicativo:



Ao executar este comando, o SSH estabelece um túnel criptografado entre a sua máquina e o servidor remoto. Você digita a senha, e pronto: já tem acesso ao shell da outra máquina. Mas apesar de ser simples, esse modo tem um problema, depender apenas de senhas é arriscado.

Login direto com endereço IP


Nem sempre você terá um domínio configurado para o servidor. Em muitos casos, o acesso será feito diretamente usando o endereço IP. Nesse caso, o comando muda apenas na parte final: em vez de `@meu-servidor.com` você usará o IP numérico.



```
ssh usuario@192.168.1.10
```

Nesse exemplo, usuario é a conta configurada no servidor e 192.168.1.10 é o endereço IP do host. Essa forma é bastante comum em redes internas ou quando o servidor ainda não tem um domínio associado.

💡 Dica: se você estiver usando uma porta diferente da padrão (22), pode acrescentar a opção -p. Por exemplo:



```
ssh usuario@192.168.1.10 -p 2222
```

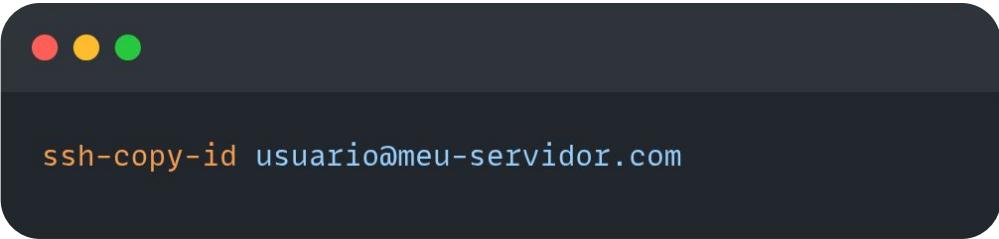
A autenticação por chaves

Uma prática bem mais segura é usar chaves criptográficas em vez de senha. Nesse modelo, você gera um par de chaves: uma fica com você (chave privada) e a outra vai para o servidor (chave pública). Assim, só quem tem a chave privada correta consegue abrir o túnel.



```
# Gerando um par de chaves RSA de 4096 bits  
ssh-keygen -t rsa -b 4096 -C "seu_email@exemplo.com"
```

Depois, é só copiar a chave pública para o servidor:

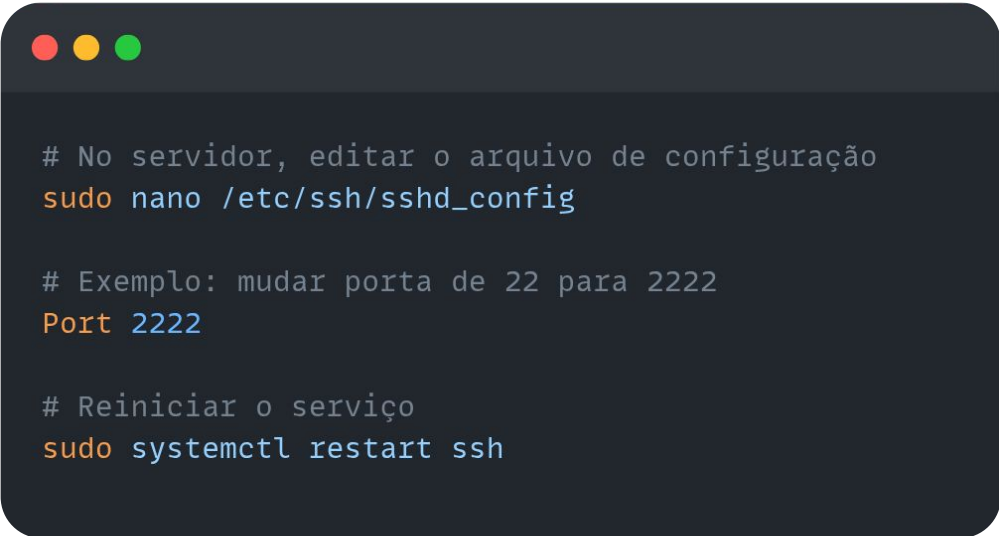


```
ssh-copy-id usuario@meu-servidor.com
```

A partir daí, o login acontece sem pedir senha, mas ainda de forma segura, porque apenas sua chave privada é capaz de autenticar o acesso.

Boas práticas de segurança

Ao longo dos anos, a comunidade percebeu que pequenas configurações fazem uma grande diferença na segurança do SSH. Uma delas é **desabilitar o login direto como root**. Isso impede que atacantes tentem adivinhar senhas do usuário mais poderoso do sistema. Outra prática comum é **alterar a porta padrão**, que normalmente é a 22. Isso não substitui segurança real, mas dificulta ataques automatizados.



```
# No servidor, editar o arquivo de configuração
sudo nano /etc/ssh/sshd_config

# Exemplo: mudar porta de 22 para 2222
Port 2222

# Reiniciar o serviço
sudo systemctl restart ssh
```

Outra dica valiosa é combinar SSH com **fail2ban** ou outros sistemas que bloqueiam tentativas de login após falhas consecutivas. Isso reduz drasticamente o risco de ataques de força bruta.

Muito além do login

Com essas práticas, o SSH se transforma em muito mais que um meio de acesso. Ele se torna uma **porta confiável para o seu servidor**. A partir dele, você pode controlar serviços, transferir dados com segurança e, principalmente, abrir caminho para recursos mais avançados como o **port forwarding**, que vamos explorar no próximo capítulo.

03

Entre portas e túneis: o poder do port forwarding

Com o SSH, não apenas abrimos portas, mas escavamos e atravessamos túneis.

Entre portas e túneis: o poder do port forwarding

Abrindo caminhos seguros dentro da rede

Depois de entender como funciona o acesso via SSH, é hora de explorar um recurso poderoso: o tunelamento. Essa funcionalidade permite redirecionar conexões de rede de um computador para outro através de um canal seguro, protegido pelo SSH. Em outras palavras, você pode “criar atalhos criptografados” entre máquinas, sem expor diretamente serviços na internet.

O port forwarding é útil em diversas situações. Imagine que você tem um banco de dados rodando em um servidor remoto que, por motivos de segurança, não pode ser acessado de fora. Com o tunelamento via SSH, é possível redirecionar essa porta para o seu computador local, como se o serviço estivesse rodando na sua própria máquina.

Port Forwarding Local

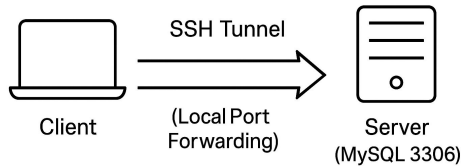
Esse é o tipo mais comum de tunelamento. Ele pega uma porta da sua máquina local e redireciona para um serviço dentro do servidor.

Por exemplo, se você quiser acessar um banco MySQL que está no servidor remoto na porta `3306`, pode usar:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The command `ssh -L 3306:localhost:3306 usuario@meu-servidor.com` is displayed in a light blue monospace font.

```
ssh -L 3306:localhost:3306 usuario@meu-servidor.com
```

Aqui, a primeira parte `3306` indica a porta no seu computador, enquanto `localhost:3306` refere-se ao serviço rodando dentro do servidor. Depois de rodar esse comando, basta abrir seu cliente MySQL local e conectar em `localhost:3306` como se o banco estivesse rodando na sua máquina.

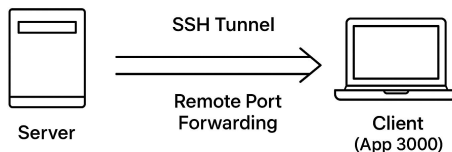


Port Forwarding Remoto

Esse método funciona no sentido inverso: você abre uma porta no servidor remoto que redireciona para um serviço local. É útil quando você precisa, por exemplo, mostrar uma aplicação em desenvolvimento rodando no seu notebook para alguém acessar via servidor remoto.

```
ssh -R 8080:localhost:3000 usuario@meu-servidor.com
```

Nesse caso, qualquer conexão feita no servidor remoto para a porta **8080** será redirecionada para o serviço que está rodando na sua máquina local na porta **3000**.



Port Forwarding Dinâmico

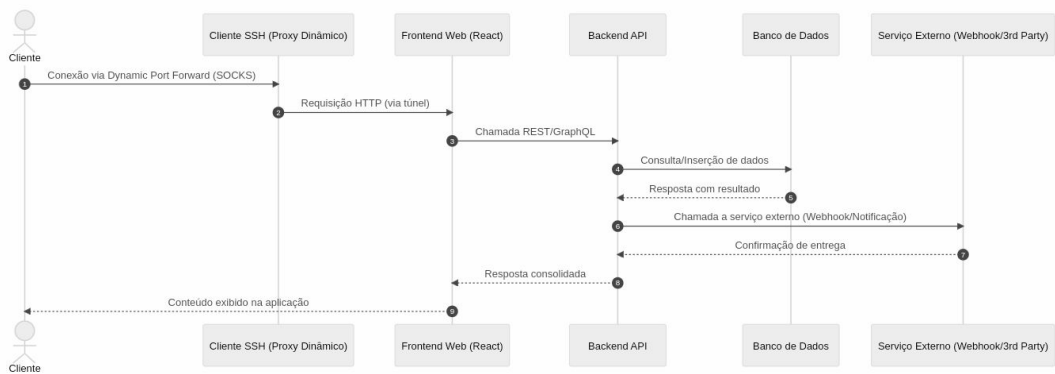
O modo dinâmico transforma sua sessão SSH em um proxy SOCKS, permitindo que você redirecione o tráfego da internet de programas locais através do servidor remoto. Esse é o truque usado, por exemplo, para navegar na web como se estivesse em outro país ou para aumentar a privacidade.

```
ssh -D 1080 usuario@meu-servidor.com
```

Depois disso, basta configurar seu navegador ou aplicativo para usar um proxy SOCKS na porta `1080` do `localhost`.

O port forwarding dinâmico se destaca por oferecer flexibilidade: em vez de redirecionar apenas uma porta específica, ele cria um proxy completo capaz de encaminhar diferentes tipos de tráfego pela conexão SSH. Isso significa que você pode navegar na web, acessar APIs, consultar bancos de dados ou até mesmo interagir com serviços externos, sempre com o tráfego encapsulado de forma segura dentro do túnel.

Um exemplo prático é quando configuramos um proxy SOCKS usando o comando `ssh -D`. A partir desse ponto, todo o tráfego do navegador ou de uma aplicação configurada para usar esse proxy passa pelo túnel, chegando ao servidor remoto antes de alcançar a internet. Na prática, é como se você estivesse navegando a partir do servidor remoto, com a vantagem de não expor sua conexão diretamente.



No diagrama que vimos, o cliente estabelece a conexão com o túnel dinâmico, e a partir daí todo o fluxo segue protegido. O frontend web se comunica com a API, a API interage com o banco de dados e pode ainda conversar com serviços externos. Para o cliente, a experiência é transparente: ele simplesmente acessa a aplicação e recebe a resposta final, sem precisar se preocupar com os caminhos percorridos pela rede.

Essa abordagem é especialmente útil em cenários que exigem **segurança reforçada, anonimização do tráfego ou acesso a serviços restritos por geolocalização**. Mais do que um recurso avançado do SSH, o port forwarding dinâmico é uma ferramenta que abre portas para um uso criativo e estratégico da rede, mostrando como um conceito aparentemente simples pode ter aplicações poderosas no dia a dia.

04

Notas Finais

Agradecimentos

Este conteúdo foi produzido como um desafio proposto para o bootcamp **Fundamentos de IA Generativa**, feito pela DIO em parceria com o Santander e Universia, ministrado pelo Felipe Aguiar que nos inspirou demais para que este conteúdo fosse produzido.

Então, é importante salientar que todo este conteúdo foi gerado por IA, sendo revisado por mim, Juliano Lopes, durante todo o processo, onde foram utilizados:

[CHATGPT] Para a criação do conteúdo, assim como a capa e as ilustrações presentes no livro.

[GOOGLE SLIDES] Para criação do layout e diagramação do conteúdo.

Também tenho muito que agradecer aos meus pais, Dona Lourdes e Seu Oscar, que sempre me estimularam a buscar constantemente aos estudos e sendo sempre afirmativos quanto minha capacidade.

Além disso, quero mencionar duas pessoas especiais, que sempre buscaram incentivar meu crescimento, que é meu antigo líder, Guilherme Gonçalves, famoso Toy, que acredito que irá contemplar este conteúdo, você sabe o quanto sou grato por me suportar em minhas muitas lamúrias, e também à Aline Roque, que apareceu de repente no meu caminho e tem me ajudado e muito no meu desenvolvimento, eternamente grato.



<https://github.com/ju-c-lobes/ebook-ssh>



<https://www.linkedin.com/in/juliano-lobes-votorantim-sp/>



Juliano Lopes