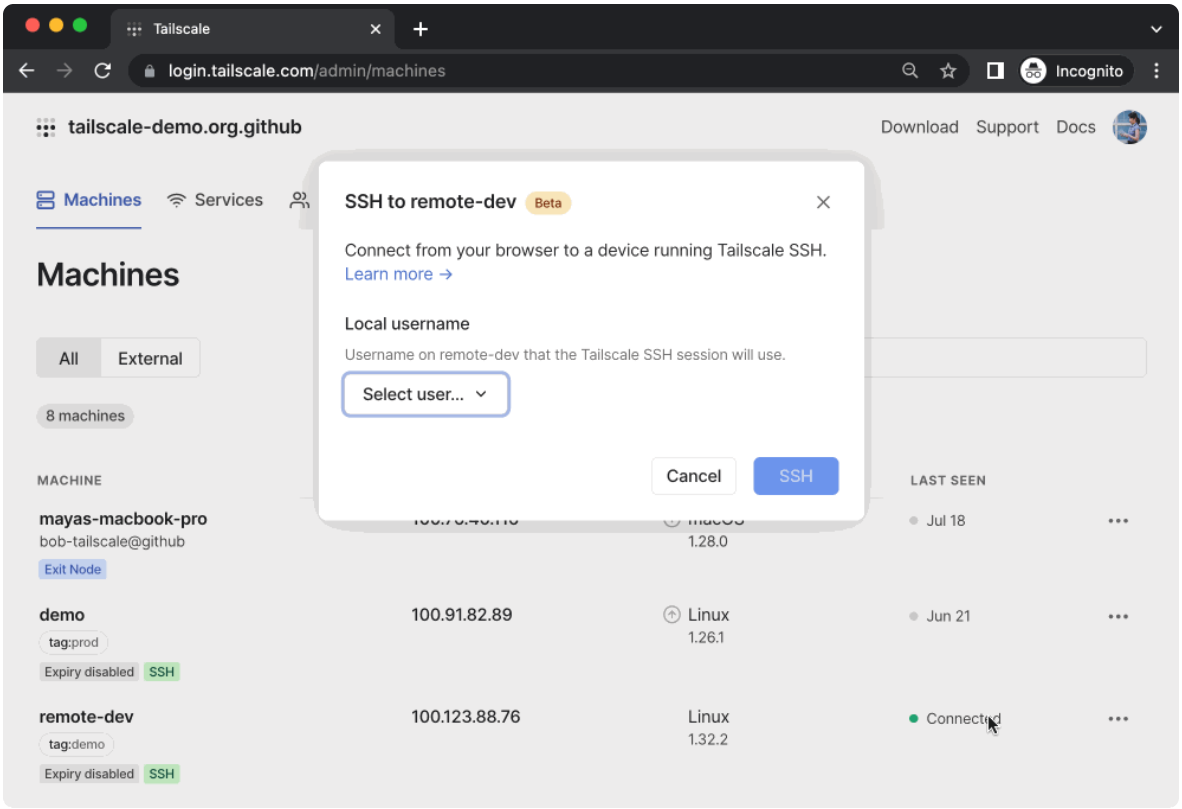
Making an SSH client the hard way

Mihai Parparita on October 27, 2022

Today, we're launching a web-based SSH client: Tailscale SSH Console.

From the Tailscale admin console, admins will now see a little "SSH…" button to connect to devices running Tailscale SSH. Click this, and you'll pop open an SSH client, right in your browser. Tailscale SSH Console is now available in beta.



To start a Tailscale SSH Console session, click "SSH" on the device, select the username you want to connect as, and reauthenticate.

Web-based SSH clients aren't new. Nearly every VPS and cloud provider already lets you connect to your VMs from the web — so how is this different? With Tailscale SSH Console, your browser *becomes* a Tailscale client, and joins your tailnet in the same way as any other device that you run Tailscale on. To make this possible, we ported the following to WebAssembly: the Tailscale client, WireGuard®, a complete userspace network stack (from gVisor), and an SSH client.

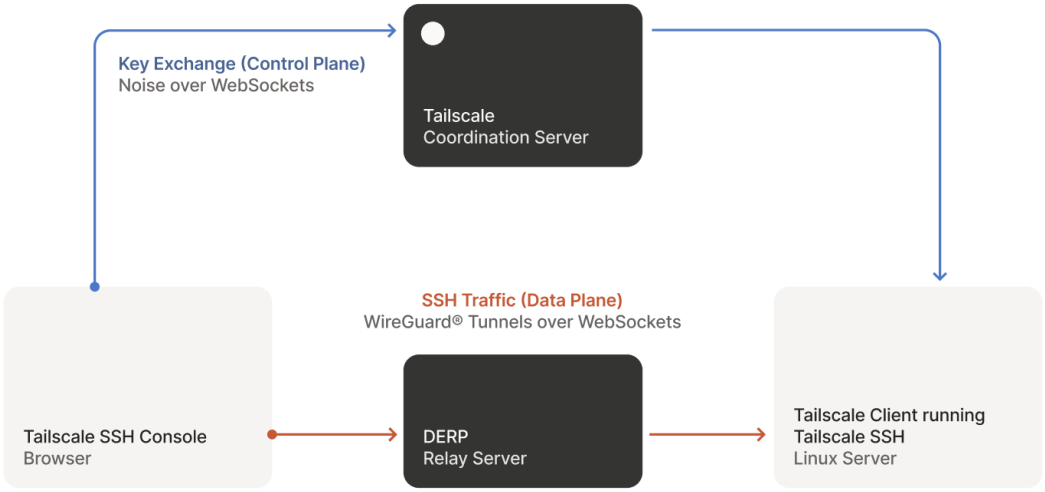
When you click the "SSH…" button, we create an ephemeral auth key in the Tailscale coordination server, and then give that auth key to a Tailscale client loaded by your browser, which starts a node with that key. The client then creates an in-memory-only

WireGuard keypair and starts to communicate with the Tailscale coordination server (to discover the rest of your tailnet) and DERP relays (to be able to connect to other nodes). Like other clients that use DERP, the traffic that passes through the relays is encrypted and not visible to Tailscale.

And because this uses Tailscale SSH, which authenticates each packet based on its WireGuard cryptographic identity, we don't need to ask you for a password, or have you upload a public key, or have you manage an `authorized_keys` file. Just click "SSH…" *and you're in* 🪄.

One year earlier

Tailscale SSH Console started out as Brad's vacation hack to see if Go's WebAssembly support was good enough to allow the Tailscale client to be compiled for the browser. After some hacking and slashing to fix build errors (including changes to wireguard-go), we had a binary that would load in the browser — but it couldn't connect to anything.



Tailscale SSH Console connects your browser to another node running Tailscale SSH.

Tailscale tries really hard to establish connections, and if a direct path is not available, it will use a DERP relay server to forward packets. However, a browser is an even more inhospitable environment than normal, and the DERP HTTP client reaching into connection internals was not compatible with the `fetch()-based HTTP transport` that ends up being used by Go when targeting WebAssembly. Like all problems in computer science, it could be solved with a layer of indirection. We added WebSockets as another transport for DERP, and then wrapped a WebSocket connection into something that's close enough to a `net.Conn` for the rest of the DERP code to use with no modifications. We were lucky that the Go WebSockets library that we use (and sponsor) already had good WebAssembly support, and only needed a small tweak. We would later use this same "tunnel it through WebSockets" approach for the coordination server protocol.

Though we had a working WebAssembly binary, it was rather…large. While this is unavoidable to some degree when building Go for WebAssembly, there was surely some low-hanging fruit. A friend had recently released Weave, a `wasm` viewer, and we

some low-hanging fruit. A friend had recently released weave, a `.wasm` viewer, and we extended it with a tree view to make it easier to see which packages and files were contributing to the binary size the most. We found a few unnecessary dependencies, and together with some build settings tweaks we were able to shave off a few megabytes. However, there's still more work to be done here.

Building a product

Just being able to have a Tailscale client run in the browser is cool in and of itself, but there was still the question of "OK, but what would you use it for?". This occurred while we were developing Tailscale SSH, and we saw a natural application — using this new capability to create a secure, in-browser, SSH client. We could have a client that let you browse through the devices on your tailnet that were accessible to you, made use of the policy to let you pick a username on the host, and integrated with check mode.

The most natural place for this SSH client would be Tailscale's admin console, but that is part of the closed source coordination server. We liked the idea of the WebAssembly client being open source, but we needed a way to reuse it in our closed source repo. We ended up making an NPM package to allow this, which was a bit of a novelty in our Go-centric world. Though this package was developed for internal use, we want to support other interesting applications — so please get in touch if you have something in mind.

Adding the ability to create SSH sessions from our admin panel led to some interesting user experience discussions. Would these sessions be transient, just open for a few minutes to check on something, or longer affairs? Would the user want to have multiple sessions running concurrently? As we pondered adding tabs to the panel, and possibly making things draggable and minimizable, it became apparent that we were effectively creating a bug-ridden, informally-specified implementation of half a window manager.

Rather than get into that business, we decided we should use real windows, and thus allow the user to have the same flexibility that they would with a real terminal. To make these windows fast, they are "child" windows that the main admin panel opens and uses as rendering surfaces – all of the logic and state still remains in the parent window. This approach has good performance characteristics, but did require some changes to our frontend dependencies. To make it easier to iterate on these interface changes, we recreated some tooling to allow us to use in-development code with our prod tailnet, so that we could try out more realistic scenarios.

Maya demonstrates how to use Tailscale SSH Console.

To try out Tailscale SSH Console: first, enable Tailscale SSH on the device you're trying to connect to. Then, from the machines tab of the admin console, select "SSH..." to connect to any machine that your access controls allow you to connect to. Re-authenticate, and you're in. See the documentation to learn more.

Share via

← Back to index

SSH Keys
Docker SSH
DevSecOps
Multicloud
NAT Traversal
IPv4 vs IPv6
MagicDNS
PAM
PoLP

Overview
Pricing
Downloads
Documentation
How It Works
Compare Tailscale
Customers
Changelog
Use Tailscale Free

Company
Newsletter
Press Kit
Blog
Careers
Contact Sales
Contact Support
Community Forum
Security
Status
Twitter
GitHub

tailscale