

Tailscale: Securing The Control Plane

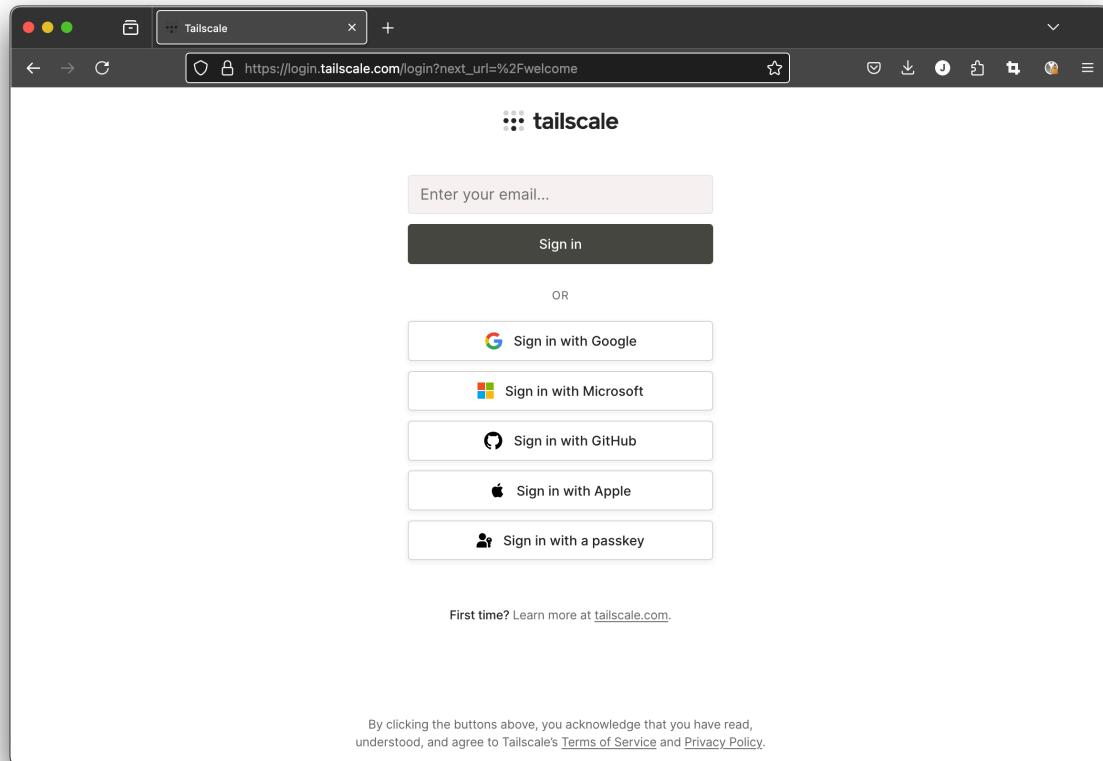
Minimizing network conceptual complexity

Before you get started

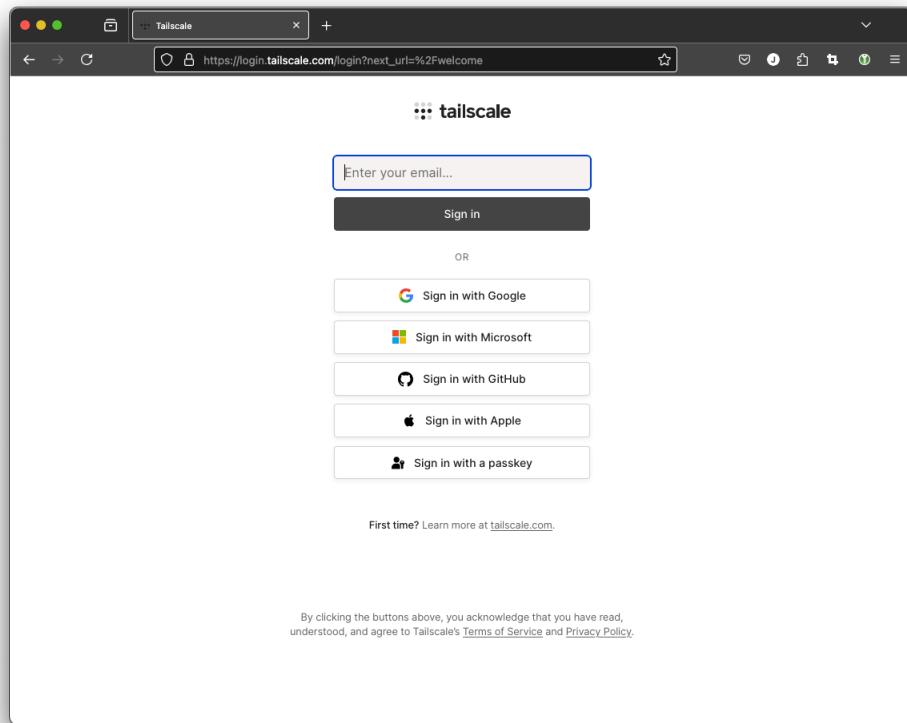
- Create a Github account
- Create or join the 'Yundera' Github organization.
- Give 'Owner' level access to the account if they need to add servers to the tailnet.

Log into Tailscale with Github

Go to <https://login.tailscale.com>

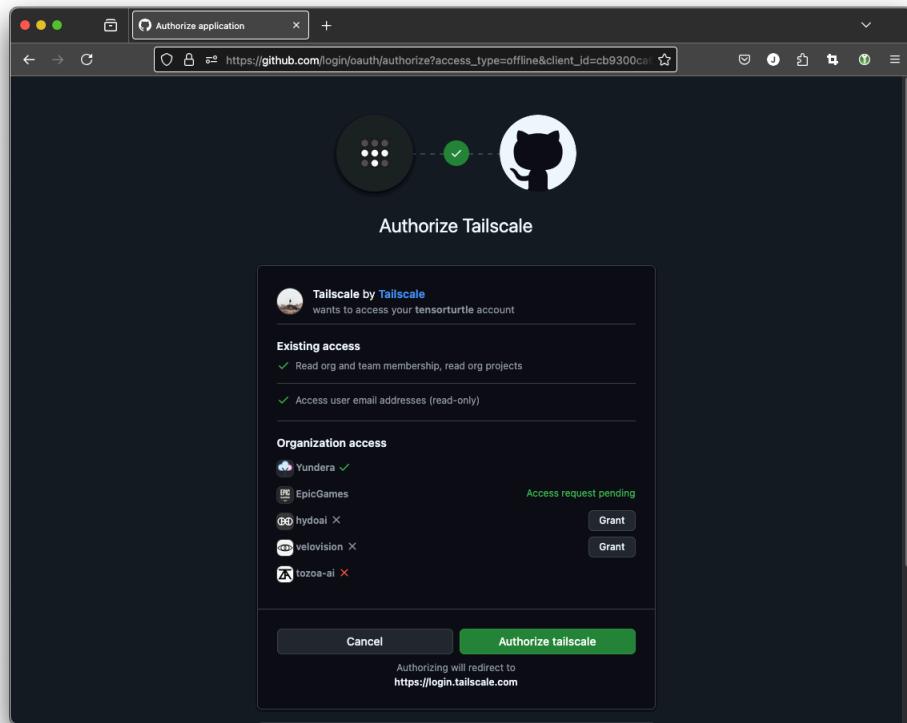


Click on 'Sign in with Github'



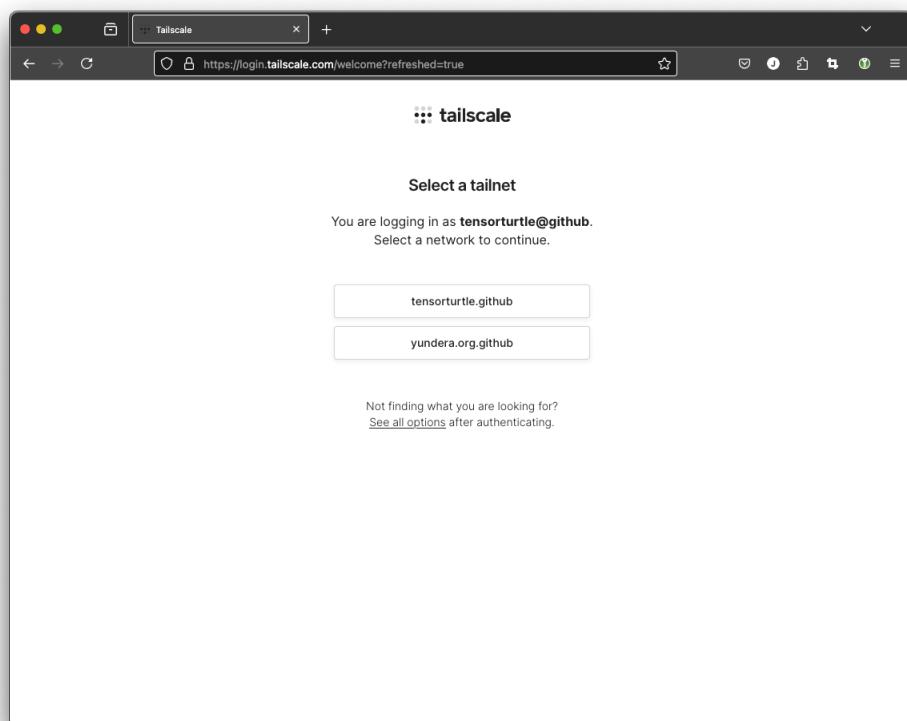
If you're the person setting up the Tailnet for the Yundera organization for the first time, make sure to 'Grant' access to the Yundera github organization.

Click on 'Sign in with Github'. On the following screen, confirm that 'Yundera' organization access is granted.



Join the Yundera Github Org's Tailnet

Next, if your Github profile is a member of the Yundera organization, you will see [yundera.org.github](https://login.tailscale.com/welcome?refreshed=true). Click on it to join.



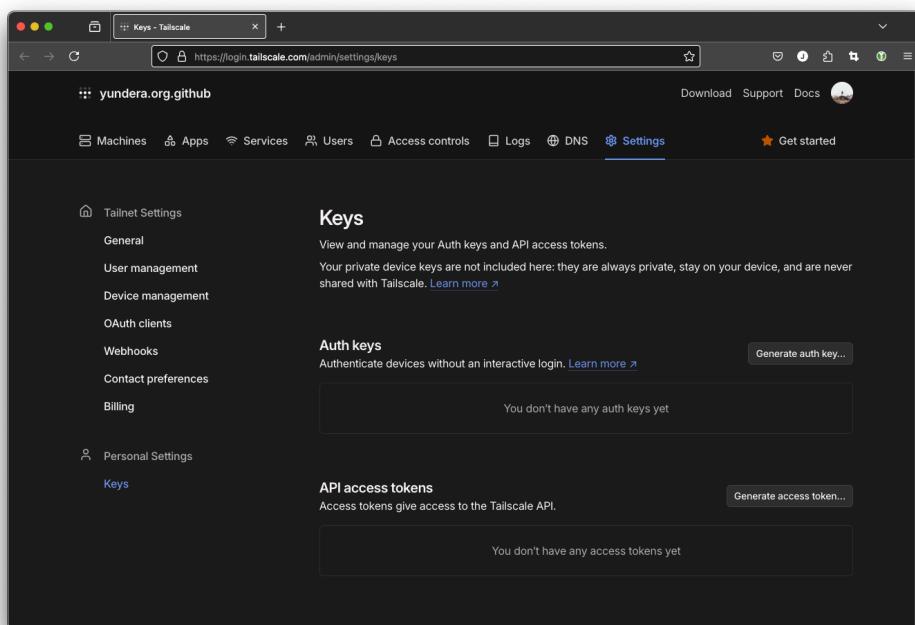
Select 'yundera.org.github' to join the virtual network that contains Yundera's servers.

Adding Machines to the Tailnet

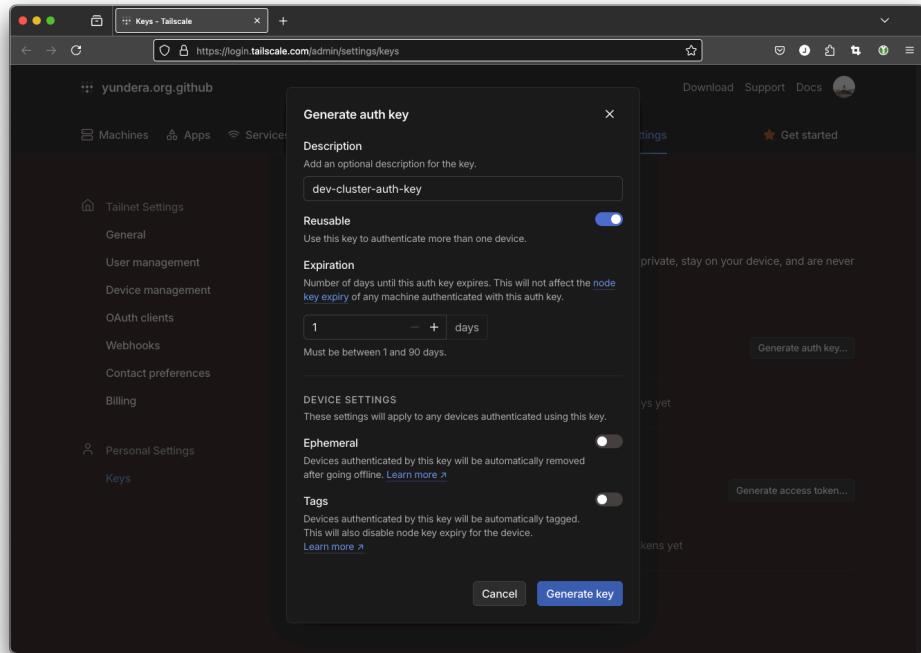
In this guide, we will add a server and a developer laptop to the tailnet.

On both the server and the developer machine, install Tailscale: <https://tailscale.com/download>.

Normally, Tailscale directs you to a web URL to authenticate when you enter [tailscale up](#). On your laptop, that's fine but for servers, since we're probably setting up multiple servers, using a single pre-generated token is probably easier. Go to <https://login.tailscale.com/admin/settings/keys> and 'Generate auth key...'



In our case, this auth key is supposed to be used once for setting up these servers and then discarded. So we set 'Reusable' to True, and set Expiration to 1 day (assuming that we'll configure all the servers within the day). Keep the Ephemeral set to False. No tags are necessary.



On Linux, the install command is:

```
curl -fsSL https://tailscale.com/install.sh | sh
```

and then

```
tailscale up --auth-key=tskey-auth-xxxxxxxxxx-XXXXXXXXXXXXXXXXXXXX
```

Control (CMD) click the link provided to authenticate. Use Github and select 'yundera.org.github' as before.

Now, two machines exist within the 'yundera.org.github' tailnet.

- 'proxmox-dev-cluster-0-node-0': A bare metal server that runs Proxmox VE
- 'jasonohn-macbook-air': Developer machine belonging to the author.

MACHINE	ADDRESSES	VERSION	LAST SEEN
proxmox-dev-cluster-0-node-0 tensorturtle@github	100.91.155.2	1.82.5 Linux 6.8.12-9-pve	Connected
jasonohn-macbookair tensorturtle@github	100.124.185.127	1.82.5 macOS 15.4.0	Connected

It is recommended to disable key expiry on remote machines such as servers. On desktops, the Tailscale app will show a pop up and remind you to do this but it's easy to forget on remote access systems and therefore recommended to disable it. This can be done from the 'Machines' page by clicking on the three dots.

At this point, we can SSH to the proxmox server using its tailnet IP: 100.91.155.2.

Preventing Access from Public IP

That's good, but our goal is to ensure that the proxmox web interface (at port 8006) is only available through the Tailnet and not through the server's public IP.

Confirm that the Proxmox VE web UI is accessible at server's public IP and port 8006. At this point, the web UI should be accessible from both the public IP and tailscale IP.

Proxmox VE supports this through the `LISTEN_IP` configuration in `pveproxy` service. Proxmox Backup Server does not have this feature - workarounds are discussed below.

https://pve.proxmox.com/pve-docs/pveproxy.8.html#pveproxy_listening_address

Find out the IPv4 tailscale address. On the server:

```
tailscale ip -4
```

Create a new file `/etc/default/pveproxy` and write a new line:

```
vi /etc/default/pveproxy
```

```
LISTEN_IP="100.ADDRESS.FROM.ABOVE"
```

Restart the relevant services

```
systemctl restart pveproxy.service spiceproxy.service
```

That's it! Confirm that the Proxmox VE web UI is running at <https://100.91.155.2:8006/> and confirm that it's NOT available at the public IP, port 8006.

Proxmox Backup Server firewall workaround

While Proxmox VE can be simply configured via `/etc/default/pveproxy` file, no such mechanism exists for Proxmox Backup Server. Therefore, we must use `iptables` to configure the network at the OS level.

```
apt install iptables-persistent
```

```
# Allow localhost access (important for the service itself)
iptables -A INPUT -i lo -p tcp --dport 8007 -j ACCEPT

# Allow from Tailscale interface
iptables -A INPUT -i tailscale0 -p tcp --dport 8007 -j ACCEPT

# Drop all other connections to port 8007
iptables -A INPUT -p tcp --dport 8007 -j DROP
```

Verify with:

```
iptables -L -n -v | grep 8007
```

It should look something like:

```
0 0 ACCEPT 6 -- lo * 0.0.0.0/0 0.0.0.0/0 tcp dpt:8007
0 0 ACCEPT 6 -- tailscale0 * 0.0.0.0/0 0.0.0.0/0 tcp dpt:8007
70 4460 DROP 6 -- * * 0.0.0.0/0 0.0.0.0/0 tcp dpt:8007
```

In this example, the last row shows 70 packets (4460 packets) dropped which means it's working.

Double-check in real life by visiting the server's public IP port 8007 (should not load), and visiting the tailscale IP port 8007 (it should work).

For API access to Proxmox, use the same tailscale IP and port. Obviously, the device from which Proxmox API is being called needs to be inside the tailnet.

Bonus: Thanks to Tailscale's 'MagicDNS', the hostname can be used instead of the IP, like so: <https://proxmox-dev-cluster-0-node-0:8006/>

Migrating Existing Proxmox VE Cluster to Tailscale

Whenever possible, it is recommended to first set up each Proxmox VE hypervisor host with Tailscale, and then afterwards create a cluster from them.

However, if you must migrate an existing cluster (that communicates with each other over public IP addresses) to tailscale, it certainly is possible.

Edit the `/etc/hosts` for each host such that the prior hostname resolves to the new tailscale IP (found with `tailscale ip -4`)

For example `/etc/hosts` on 'proxmox-dev-cluster-0-node-0':

```
127.0.0.1 localhost
#163.172.68.57 proxmox-dev-cluster-0-node-0 proxmox-dev-cluster-0-node-0
100.91.155.2 proxmox-dev-cluster-0-node-0.bone-delta.ts.net proxmox-dev-cluster-0-node-0

::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

and `/etc/hosts` on 'proxmox-dev-cluster-0-node-1':

```
127.0.0.1 localhost
#163.172.68.59 proxmox-dev-cluster-0-node-1 proxmox-dev-cluster-0-node-1
100.112.162.15 proxmox-dev-cluster-0-node-1.bone-delta.ts.net proxmox-dev-cluster-0-node-1

::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

The internals of Proxmox aren't discussed in detail here but essentially (potentially incorrectly), this modification causes the given hypervisor host to 'broadcast' out that its self-referential IP is now the new tailscale IP, and the cluster can pick this up and therefore other hypervisors will use the new tailscale IP associated with this host.