

Standing up the CA

This is the identity root for every machine in the environment. Everything else in the book depends on it existing and being trusted. Treat its keys with the same care as any other root secret.

Initialize step-ca

On the control-plane host, initialize the CA:

```
sudo curl -fsSL https://packages.smallstep.com/keys/apt/repo-signing-key.gpg -o
/etc/apt/trusted.gpg.d/smallstep.asc && echo 'deb [signed-
by=/etc/apt/trusted.gpg.d/smallstep.asc] https://packages.smallstep.com/stable/debian debs
main' | sudo tee /etc/apt/sources.list.d/smallstep.list

sudo apt update && sudo install step-cli step-ca

sudo step ca init
```

Make sure port 9000 is open on your firewall, this is the default port step-ca listens on.

The prompts and the choices made here:

Prompt	Value	Why
Deployment type	Standalone	A single self-contained CA; no need for the more complex modes at this scale.
PKI name	(your CA name)	Labels the root and intermediate.
DNS / IP for the CA	(internal CA host IP)	The address hosts will reach the CA at, inside the private network only.
CA bind address	(internal IP):9000	step-ca listens here for issuance requests.
First provisioner name	admin-provisioner	The identity allowed to request certificates. Referenced later when issuing.
Password for the CA keys	(strong secret)	Encrypts the CA's private keys at rest. Store it in the password manager.

`step ca init` produces the root certificate, the intermediate certificate, and the signing keys. The root is the thing every host will be told to trust.

The CA key password goes in a file the service reads at startup:

```
sudo nano /root/.step/secrets/password.txt
```

This file, and the key material it unlocks, are the crown jewels for your CA. Anyone who can read both can mint certificates the entire environment trusts. Lock down the host accordingly.

Set the issuance policy

Short-lived certificates are one of the main arguments for a private CA, and the issuance policy is where you enforce them. Edit the CA config:

```
sudo nano /root/.step/config/ca.json
```

Add a claims block setting the minimum, maximum, and default certificate lifetimes:

```
"claims": {  
  "minTLSCertDuration": "1h",  
  "maxTLSCertDuration": "2160h",  
  "defaultTLSCertDuration": "720h"  
}
```

What this says: a certificate can live no less than an hour, no more than 2160 hours (90 days), and defaults to 720 hours (30 days) if a duration isn't specified. The 90-day ceiling is the important one, it means a stolen certificate is worthless within a quarter no matter what, and in practice renewal happens far more often than that. Setting a *maximum* is what stops a convenient long-lived cert from quietly becoming a long-lived liability.

Capture the root fingerprint

Every host that joins the trust domain will pin the root by its fingerprint. Get it:

```
sudo step certificate fingerprint /root/.step/certs/root_ca.crt
```

Record this. It's used on every host bootstrap on the next page, and pinning by fingerprint means a host checks the CA against a value you already know, so a substituted or malicious root is rejected

instead of trusted.

Run it as a service

Run step-ca under systemd so it starts on boot and restarts on failure:

```
[Unit]
Description=Smallstep Certificate Authority
After=network-online.target

[Service]
User=root
Environment=STEPPATH=/root/.step
ExecStart=/usr/bin/step-ca /root/.step/config/ca.json --password-file
/root/.step/secrets/password.txt
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

The `--password-file` is what lets it start unattended, it reads the key password from the file rather than prompting. That convenience is also why that file's permissions matter so much: it is the thing standing between "a service that restarts cleanly" and "anyone who reads this file owns the CA."

Revision #5

Created 2026-07-20 19:35:31 UTC by Chris Landis

Updated 2026-07-27 14:50:57 UTC by Chris Landis