

Architecture

Before the configuration, the model. Three ideas make the rest of this book make sense: where the CA lives, what the trust chain looks like, and the fact that every internal connection uses *two* certificates, not one.

Where the CA lives

The certificate authority runs on an internal control-plane host, reachable only from inside the private network. It is never exposed publicly. Its only job is to answer certificate requests from hosts that have already proven they belong in the environment, and to hold the signing key that every host trusts. Because that key is the root of all internal trust, the CA host is treated as one of the most sensitive systems in the environment, on par with the identity provider and the secrets manager.

The trust chain

Step-ca is initialized with a **root** certificate and an **intermediate** certificate. The root signs the intermediate; the intermediate signs the certificates issued to hosts. Every host in the environment is configured to trust the root. From then on, a certificate is trusted if it chains back to that root; root signed the intermediate, intermediate signed the host cert. This is why nginx is configured with `ssl_verify_depth 2` later on: the chain is two links deep, and verification has to be allowed to trace both.

Trust is established once, by pinning the root's fingerprint when a host bootstraps (covered on the next page). After that, no host ever has to be told about individual certificates, it trusts anything the CA signs, and distrusts everything else.

The two-certificate model

A single mutually-authenticated connection involves **two certificates**, one presented by each end.

- The **WAF**, or other service, presents a **client certificate** identifying itself. This proves to the backend that the caller is the authorized service and not some other host that happens to be on the network.

- The **backend** presents a **server certificate** identifying it as `service.int.example`. This proves to the service, such as a (WAF), that it reached the right backend and not an impostor.

Each end verifies the other's certificate against the shared CA root. If either certificate is missing, expired, or not signed by the CA, the connection is refused. That mutual check is the whole point. The backend won't serve just anyone who can reach it, and the service won't forward to just any host claiming to be the backend.

Revision #4

Created 2026-07-20 19:35:08 UTC by Chris Landis

Updated 2026-07-27 14:56:15 UTC by Chris Landis