BUOYANT'S
SERVICE MESH
ACADEMY

The creators of
LINKERD

Certificate Management with Linkerd

Flynn, Technical Evangelist for Linkerd

@flynn (slack.linkerd.io)

@BuoyantIO

@kflynn



What's on the agenda?



Workshop with a bit of background first!

- → Intro to certificates and certificate hierarchies
- → How identity and certificates work together for Linkerd
- → The tradeoffs for certificate rotation
- → Hands-on workshop!

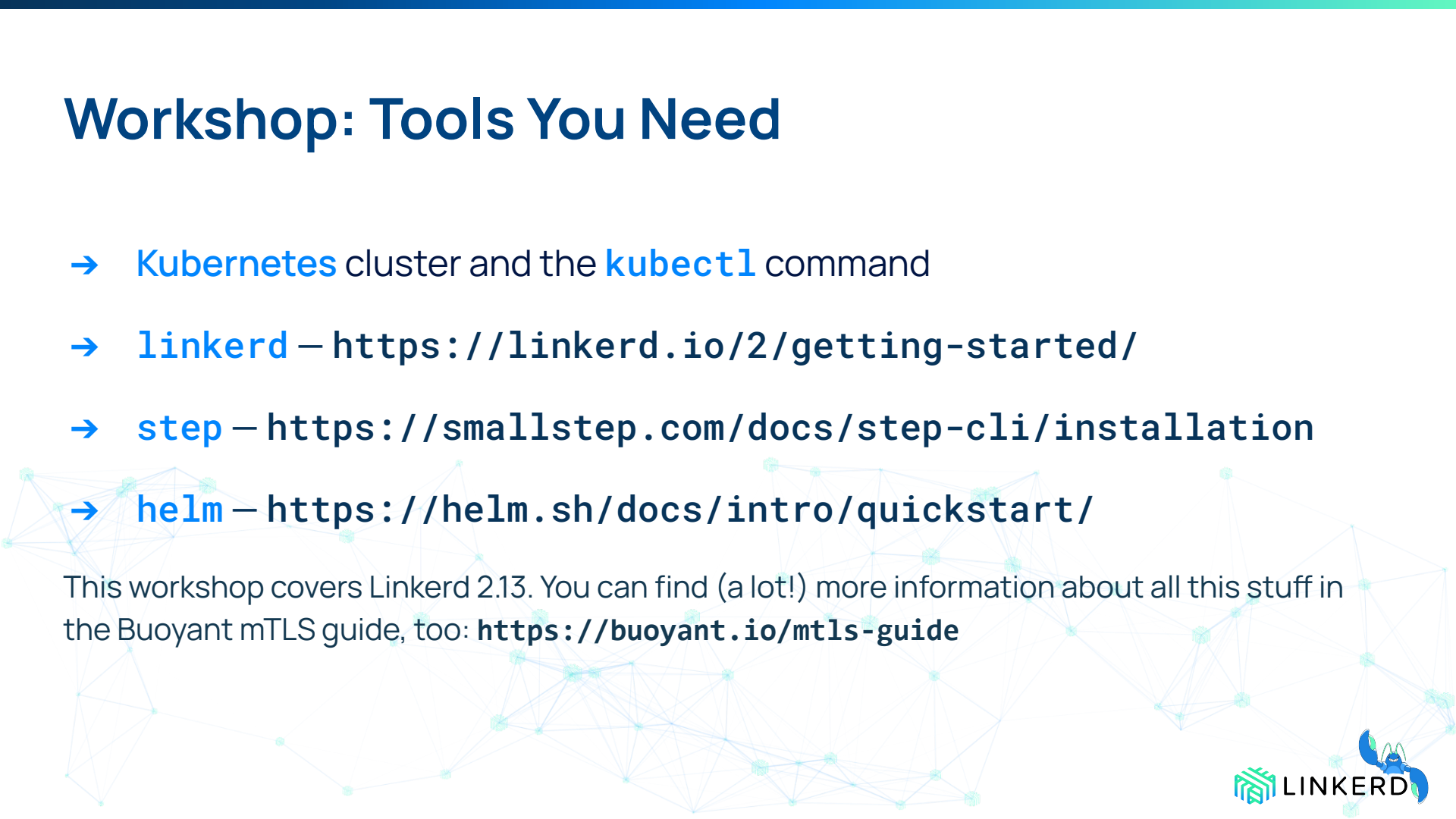


LINKERD

Workshop: Tools You Need

- ➔ Kubernetes cluster and the `kubectl` command
- ➔ `linkerd` — **`https://linkerd.io/2/getting-started/`**
- ➔ `step` — **`https://smallstep.com/docs/step-cli/installation`**
- ➔ `helm` — **`https://helm.sh/docs/intro/quickstart/`**

This workshop covers Linkerd 2.13. You can find (a lot!) more information about all this stuff in the Buoyant mTLS guide, too: **`https://buoyant.io/mtls-guide`**



BUOYANT

Creators of LINKERD

Join Buoyant's Linkerd Forum!

Searchable, long-lived help threads and technical content.
(Slack is great, but messages die after 90 days.)
linkerd.buoyant.io



BUOYANT

SERVICE MESH ACADEMY

Creators of LINKERD



Fundamentals of the Service Mesh Online Course + Certification

With hands-on LINKERD labs at learn.buoyant.io











Enterprise early access

We're adding a few more **early access slots** for our enterprise Linkerd distro.

- ➔ Simplified multi-cloud & multi-cluster management
- ➔ SLAs around CVE remediation
- ➔ Cost-aware routing and automated health checking and alerting
- ➔ Fully automated, gitops-compatible installation and upgrades
- ➔ Enterprise-only features, and 24x7 support

Send William a DM on the Linkerd Slack



BUOYANT

Creators of LINKERD

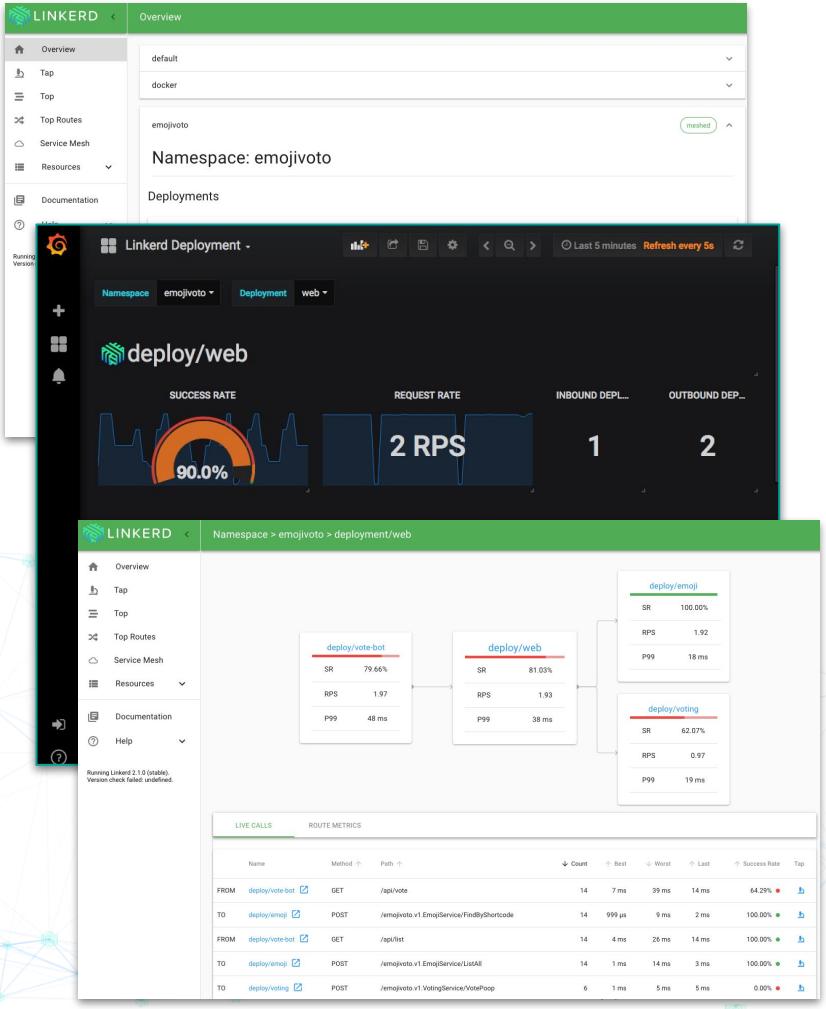
Background: Linkerd

➔ **Security:** Transparent mTLS, cert management and rotation, policy

➔ **Reliability:** Retries, timeouts, load balancing, circuit breaking*

➔ **Observability:** Service-level *golden metrics*: success rates, latencies, throughput. Service topologies.

Linkerd focuses on **operational simplicity** first and foremost; it's very lightweight and fast.



Background: Public-key Cryptography

Public-key (asymmetric) cryptography forms the basis of trust in Linkerd

- Instead of "a key" use a keypair: a public key and a private key
  - What the public key encrypts, the private key decrypts
  - What the private key encrypts, the public key decrypts
  - Given just the public key, recovering the private key is "impossible"
- If Alice encrypts something with Bob's public key, only Bob can read it…
  - …because Bob is the only one with Bob's private key
- If Alice encrypts something with her own private key, only she could have written it…
  - …because Alice is the only one with Alice's private key
  - (This is very common in practice: encrypting a hash of some data with one's private key is called signing it.)

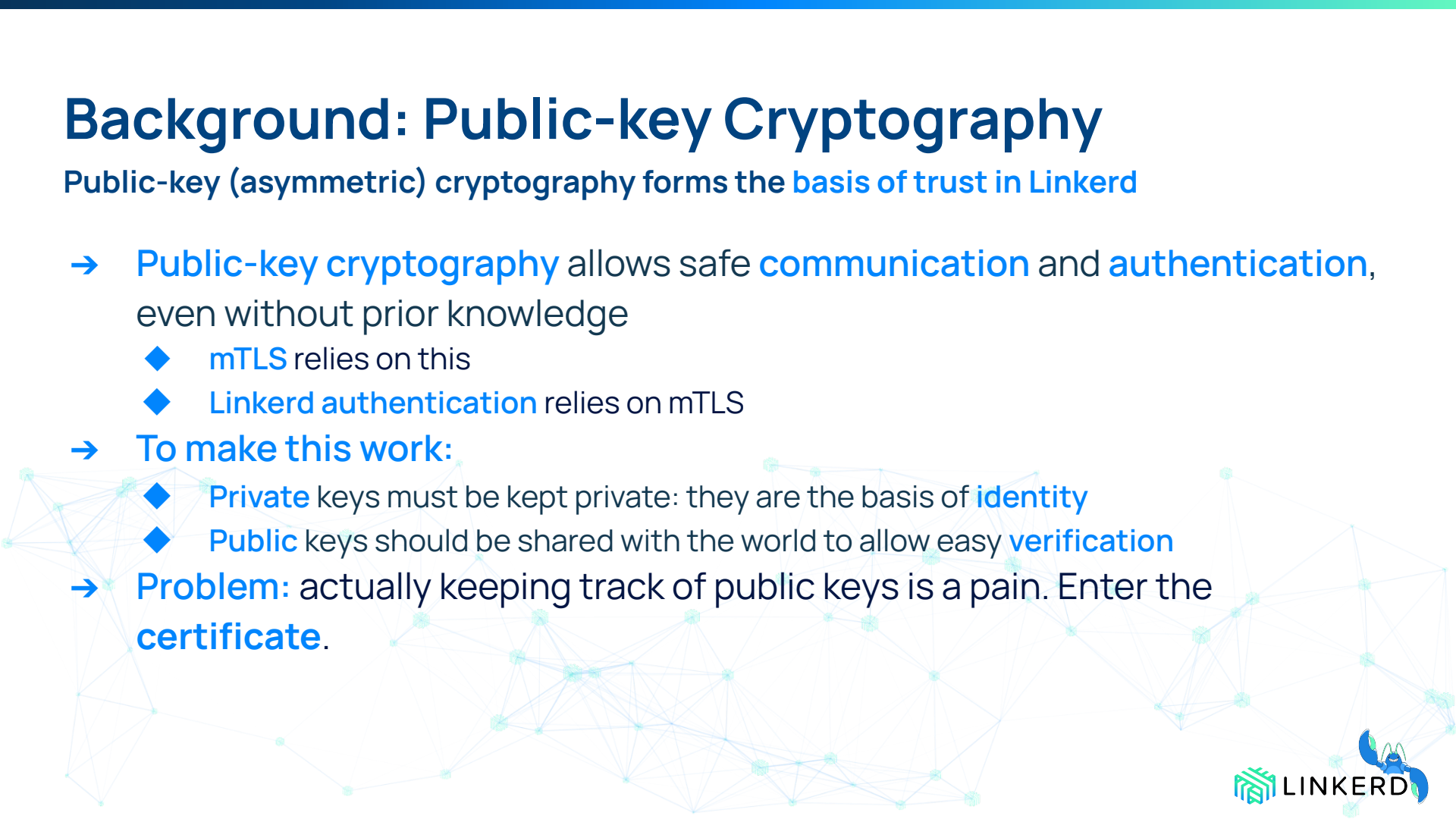


LINKERD

Background: Public-key Cryptography

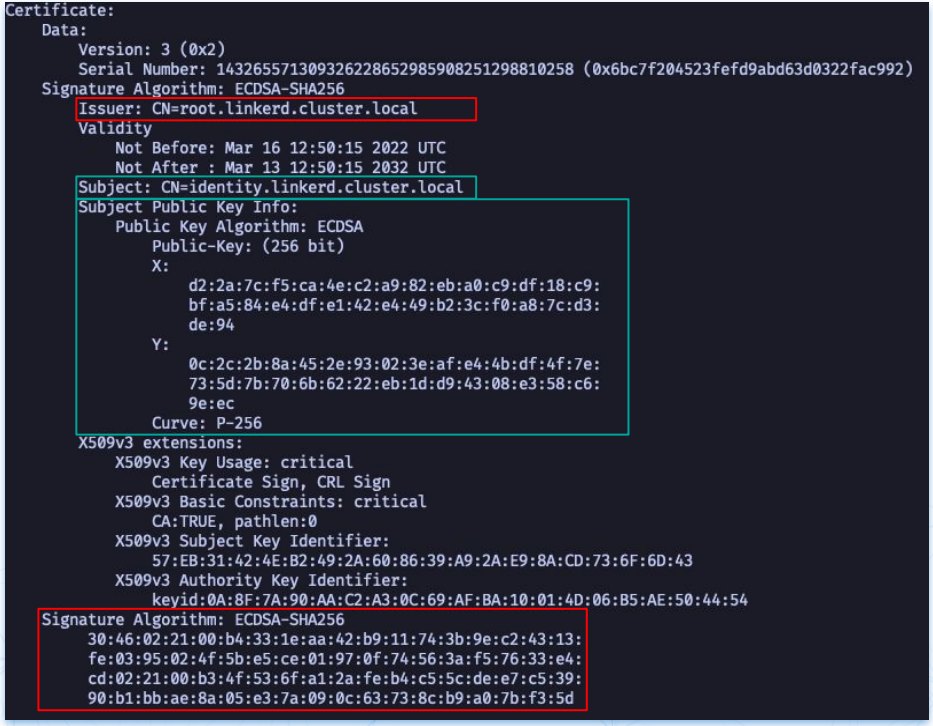
Public-key (asymmetric) cryptography forms the basis of trust in Linkerd

- ➔ Public-key cryptography allows safe communication and authentication, even without prior knowledge
  - mTLS relies on this
  - Linkerd authentication relies on mTLS
- ➔ To make this work:
  - Private keys must be kept private: they are the basis of identity
  - Public keys should be shared with the world to allow easy verification
- ➔ Problem: actually keeping track of public keys is a pain. Enter the certificate.



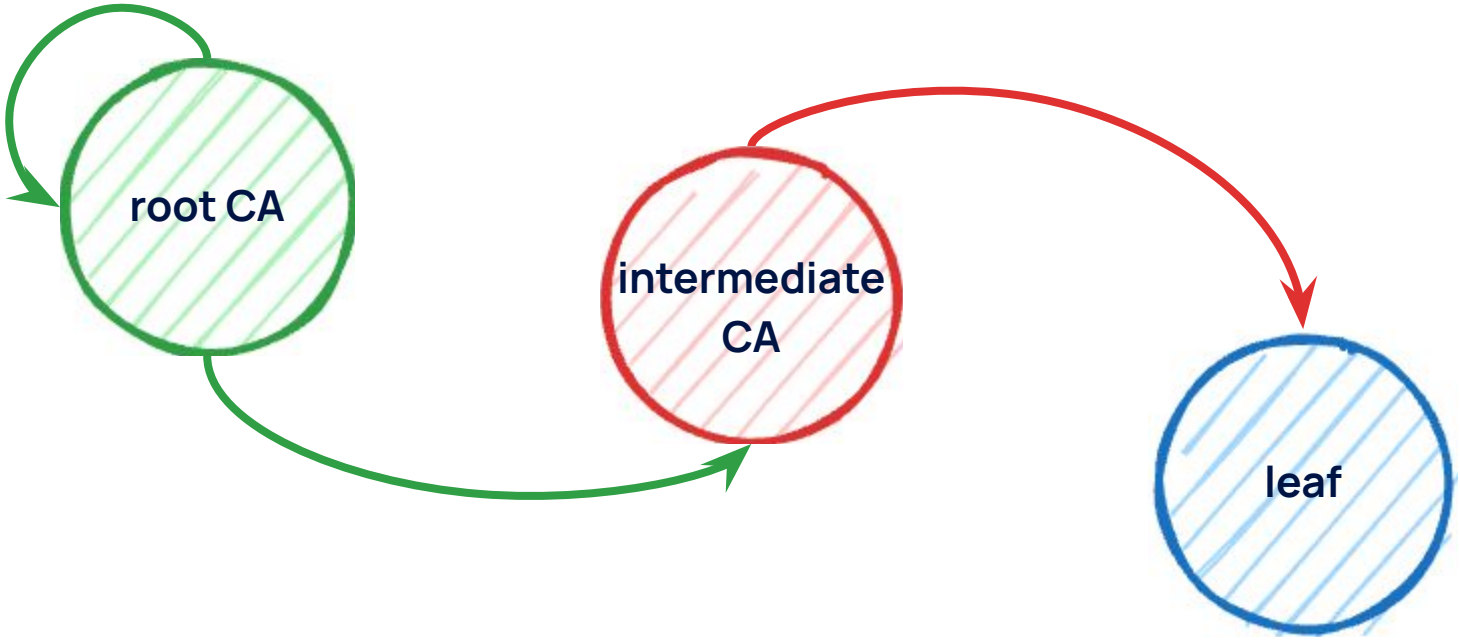
Background: Certificates

- X.509 certificates are data structures that bind a *subject* (really a name) to a keypair.
- Only public information is in a certificate – they're safe to share.
- Certificates are signed to prevent them from being spoofed, so they can be trusted.
- Rotation regenerates the keypair so that you're not relying on any one key for too long.



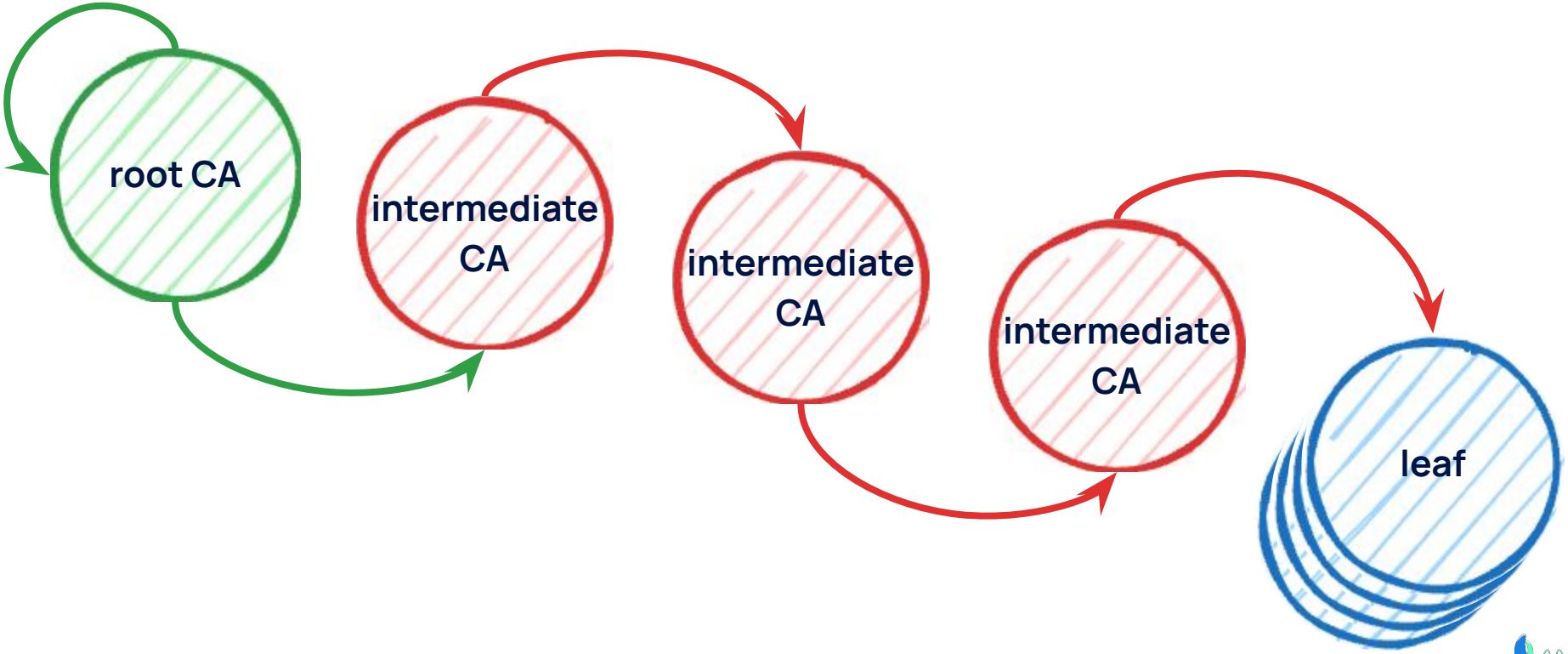
LINKERD

Certificate Hierarchy



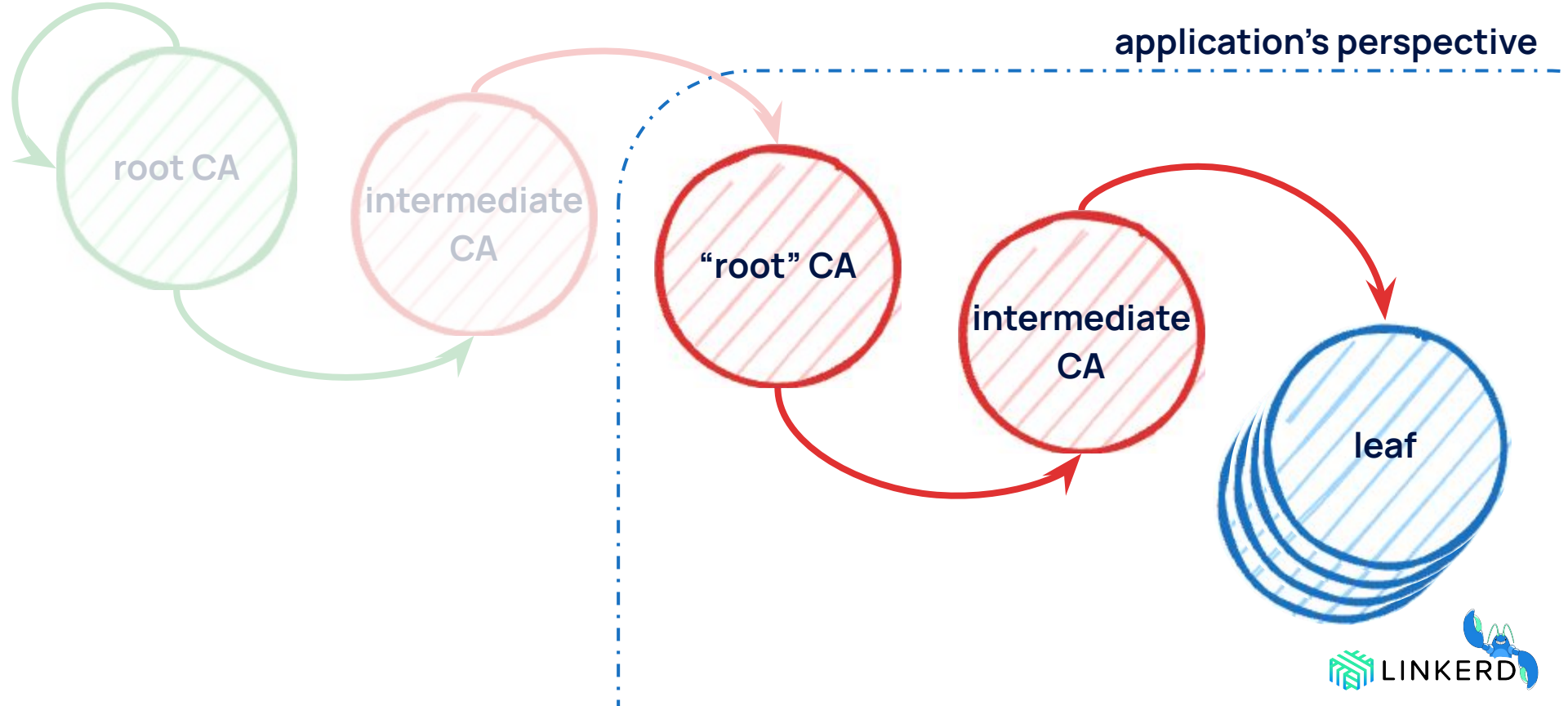
LINKERD

Certificate Hierarchy

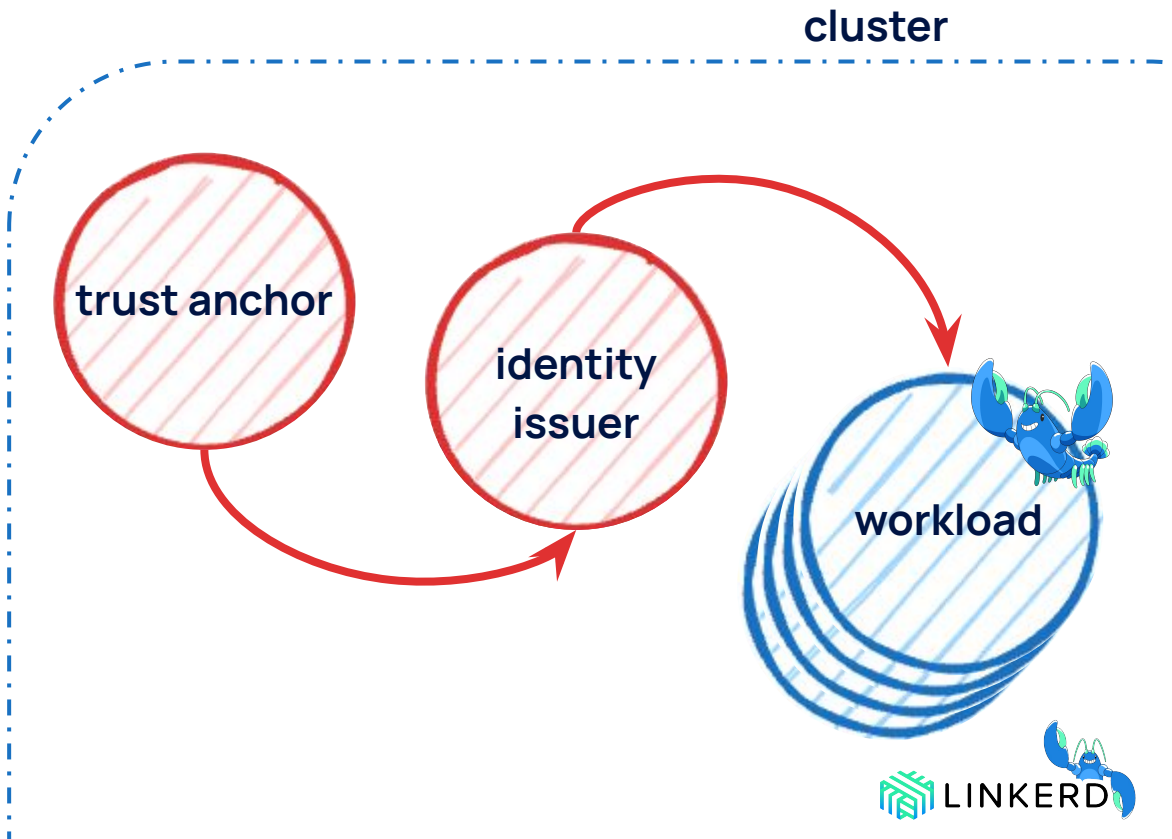


LINKERD

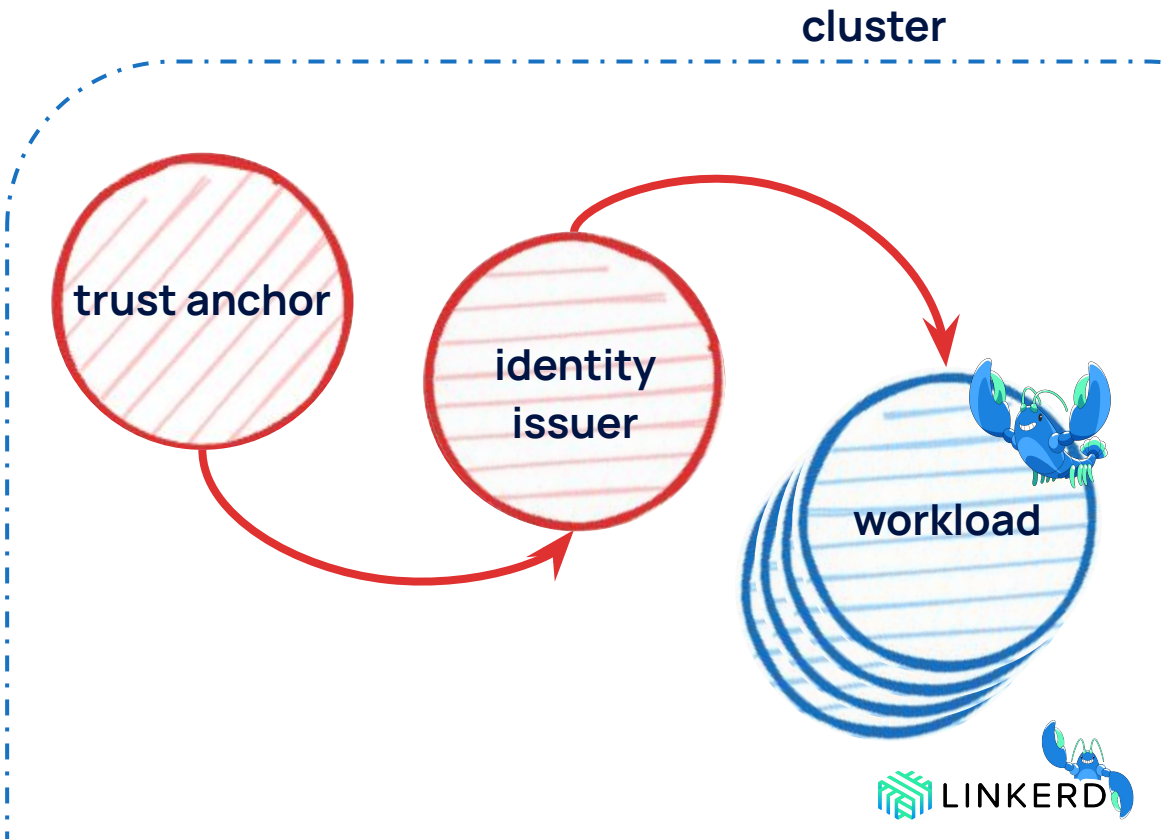
Certificate Hierarchy: Delegation



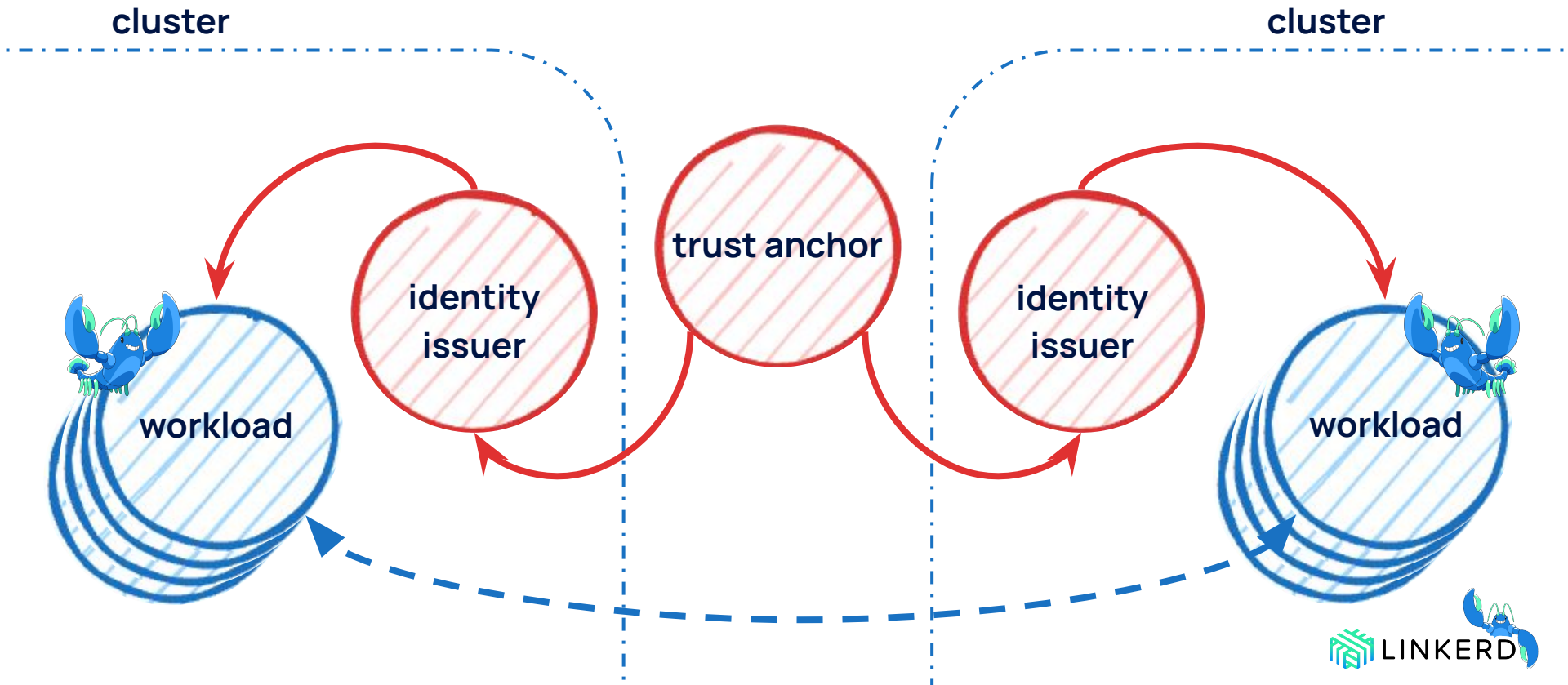
Certificate Hierarchy in Linkerd



Certificate Hierarchy in Linkerd: Rotation



Certificate Hierarchy in Linkerd: Multicluster

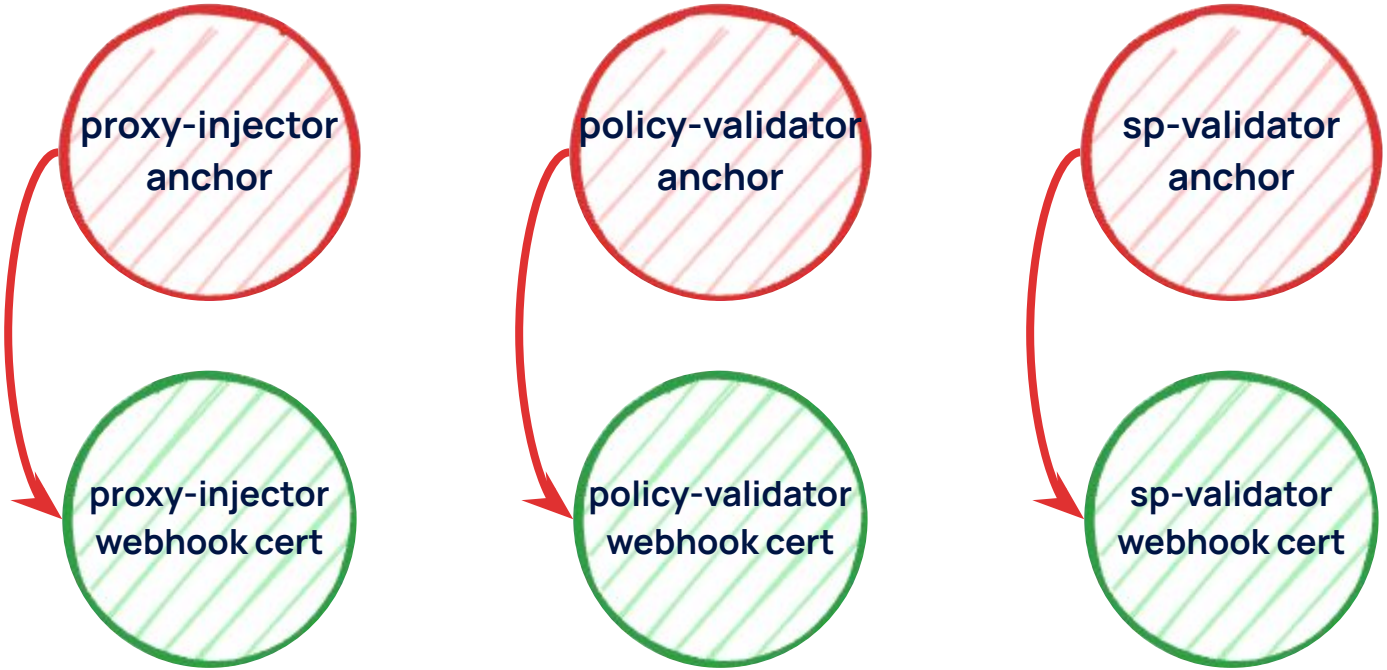


Certificate “Hierarchy”: Webhooks



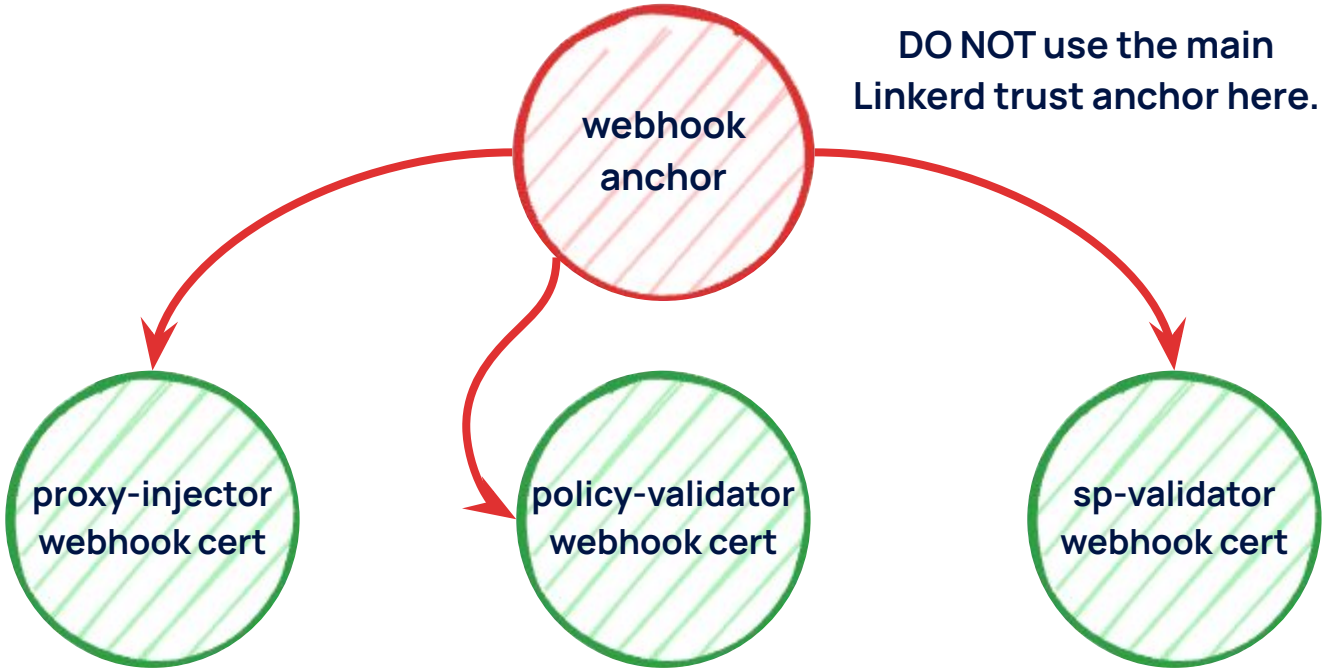
LINKERD

Certificate Hierarchy: Webhooks



LINKERD

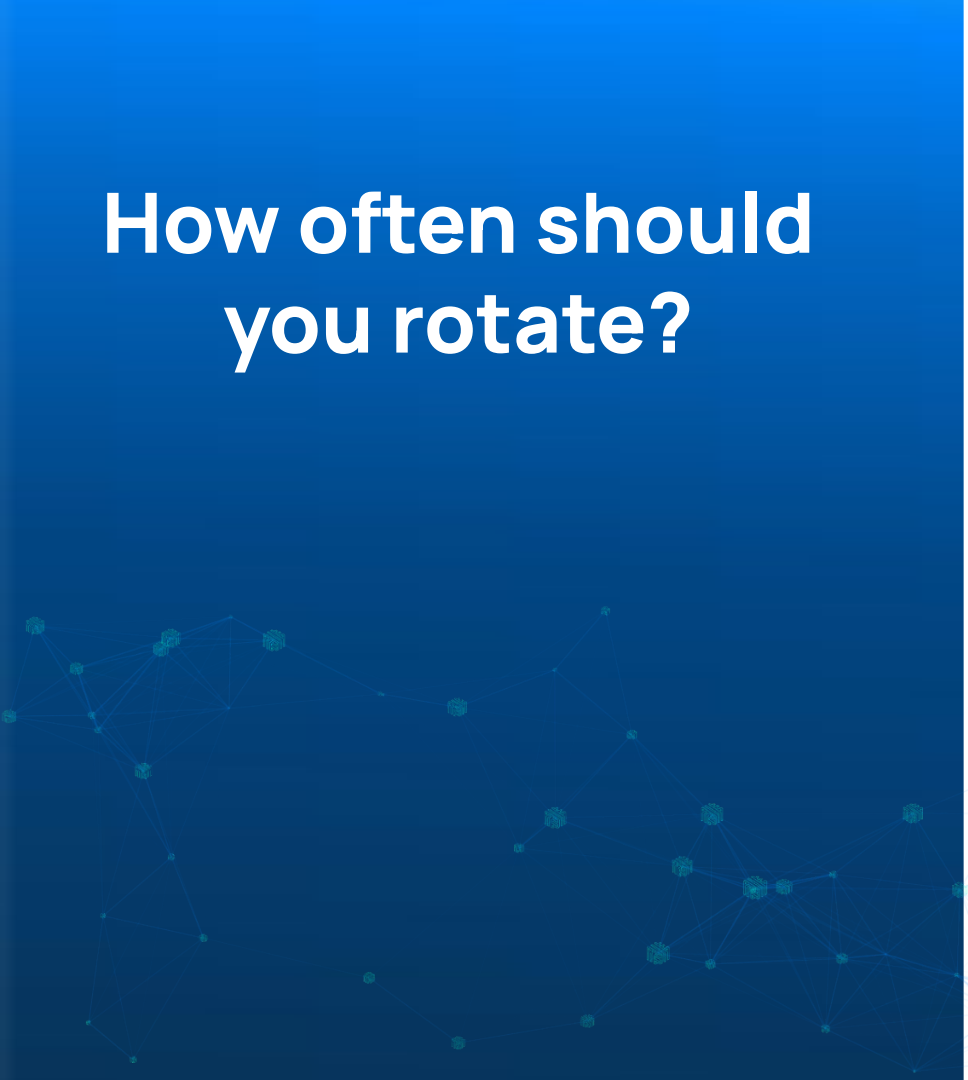
Certificate Hierarchy: Webhooks



LINKERD

How often should you rotate?

- Rotating more quickly limits risk:
  - If cert is compromised, it's going to be compromised until the next rotation (at least)!
  - Re-establishing identity every few hours **greatly** limits risk of a compromised certificate being used against you.
  - You **must** use your rotation practices and mechanisms to make sure that they actually work!
- Longer periods make it easier to experiment and focus on your work, not on your platform.



Workshop!

Follow along! **https://github.com/BuoyantIO/service-mesh-academy**
(in the **l5d-certificate-management** directory)

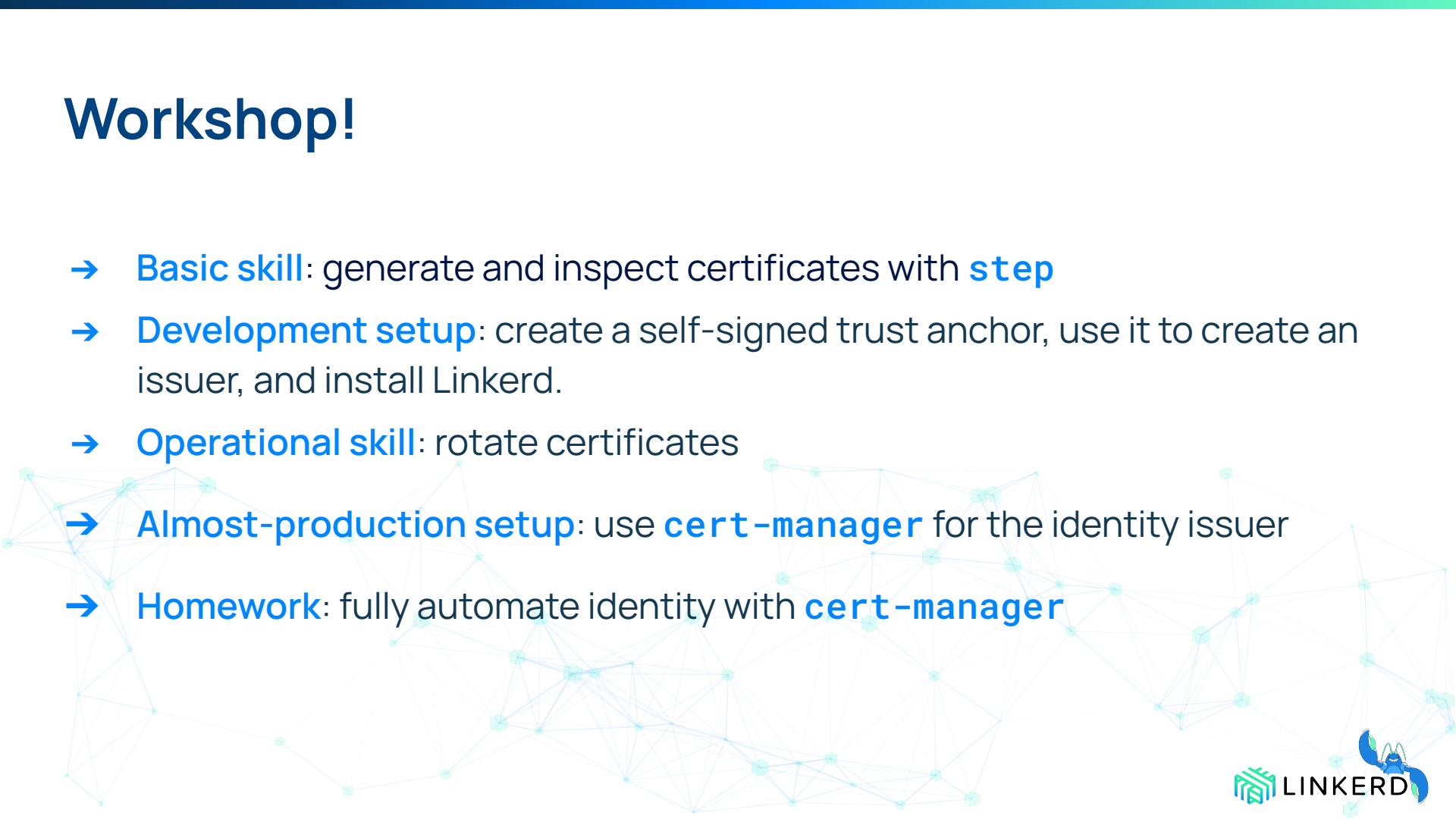
Tools we'll use:

- ➔ Kubernetes cluster and the **kubectl** command
- ➔ **linkerd** — **https://linkerd.io/2/getting-started/**
- ➔ **step** — **https://smallstep.com/docs/step-cli/installation**
- ➔ **helm** — **https://helm.sh/docs/intro/quickstart/**



Workshop!

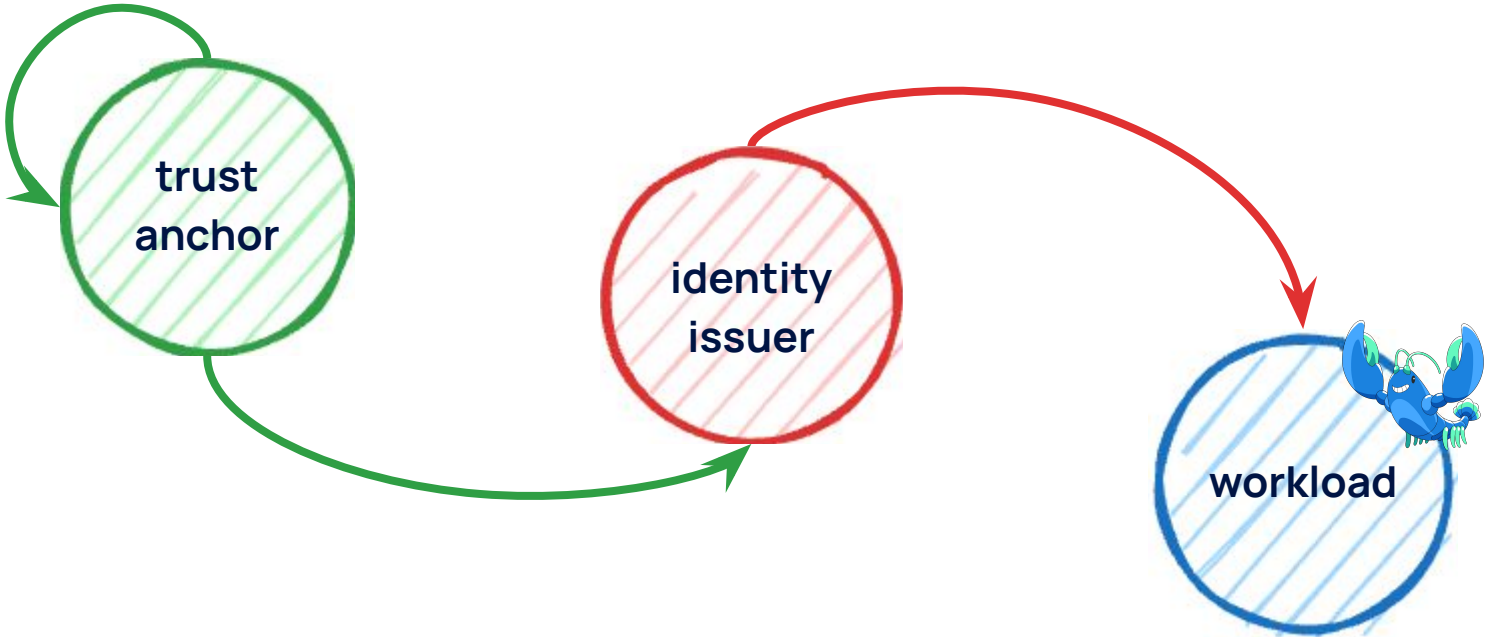
- ➔ Basic skill: generate and inspect certificates with `step`
- ➔ Development setup: create a self-signed trust anchor, use it to create an issuer, and install Linkerd.
- ➔ Operational skill: rotate certificates
- ➔ Almost-production setup: use `cert-manager` for the identity issuer
- ➔ Homework: fully automate identity with `cert-manager`



DEMO



Development Setup



LINKERD

Repo for this demo

github.com/BuoyantIO/service-mesh-academy.git

BUOYANT

SERVICE MESH ACADEMY

Creators of LINKERD

Monthly hands-on, engineer-focused training from the creators of the service mesh

August 17: What's Coming in Linkerd 2.14 (with William Morgan)

SIGN UP TODAY!
buoyant.io/sma



BUOYANT'S
SERVICE MESH
ACADEMY

The creators of
LINKERD

Thanks so much!

Flynn
Tech Evangelist, Buoyant
flynn@buoyant.io

