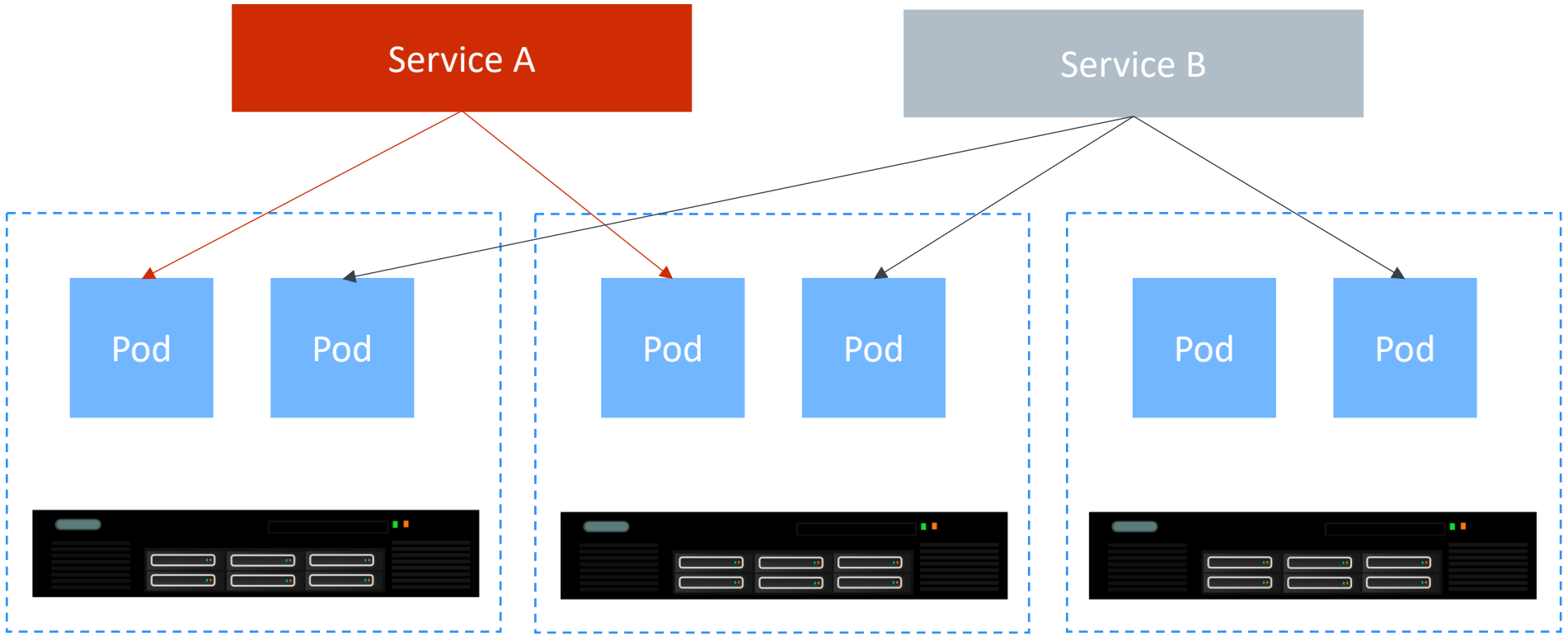
OAuth 2 in Kubernetes

Neil Madden

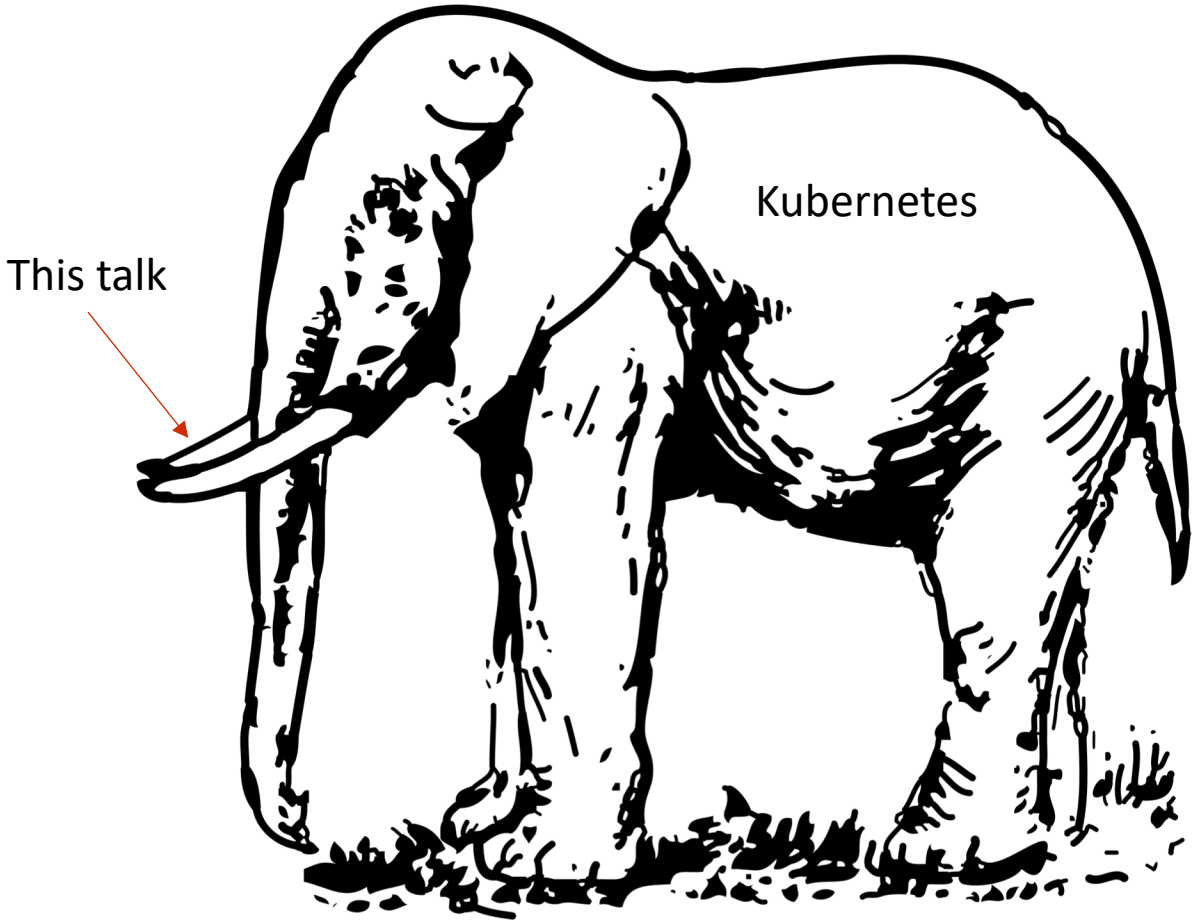
OAuth Security Workshop, Stuttgart 2019

FORGEROCK

Kubernetes


Service A
Service B
Pod
Pod
Pod
Pod
Pod
Pod
Pods (Containers)
Nodes

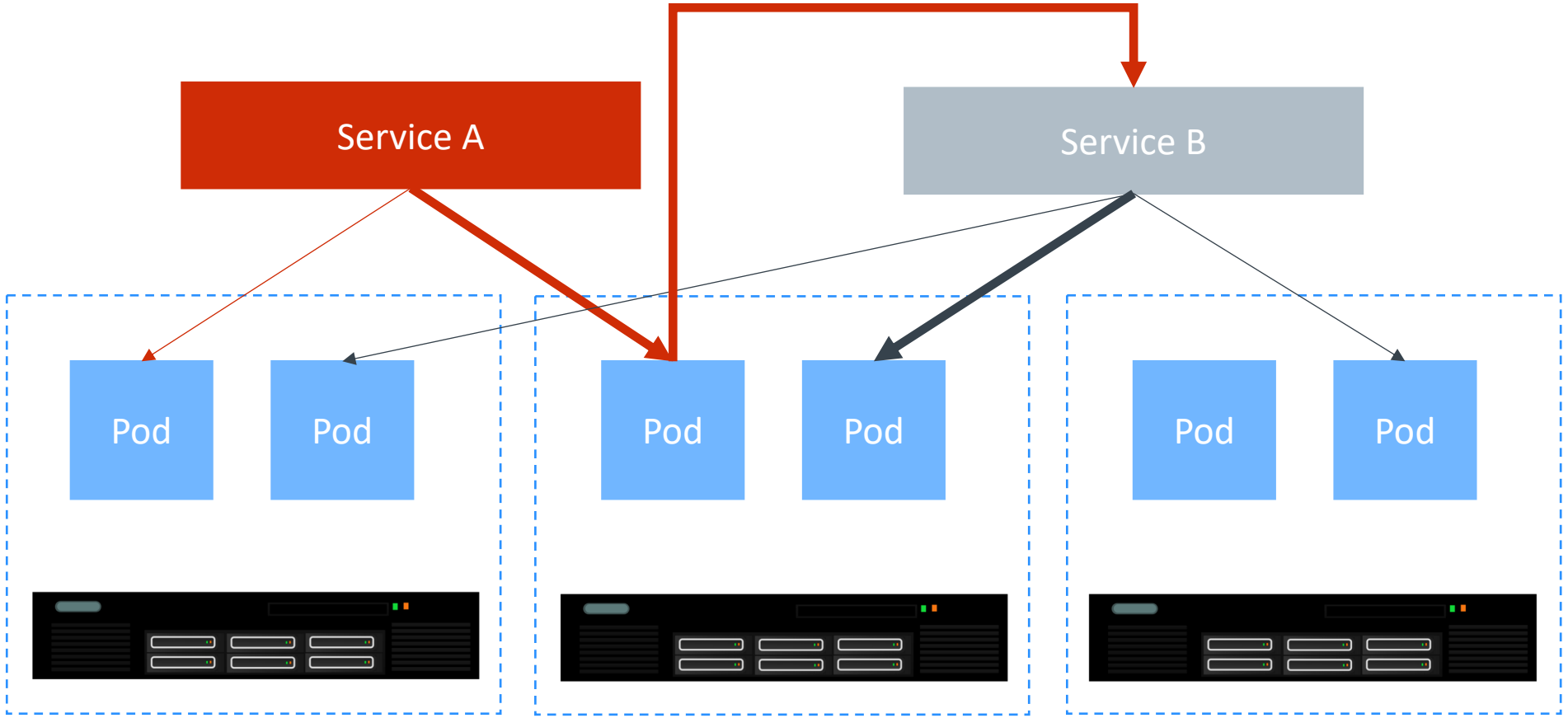
Focus of this talk



- Using OAuth to authorize *service to service* calls in Kubernetes
- Bootstrapping a secure system
- Not covering:
  - User authorization
  - Ingress
  - Lots of other details!

FORGEROCK

Service to service (normal)



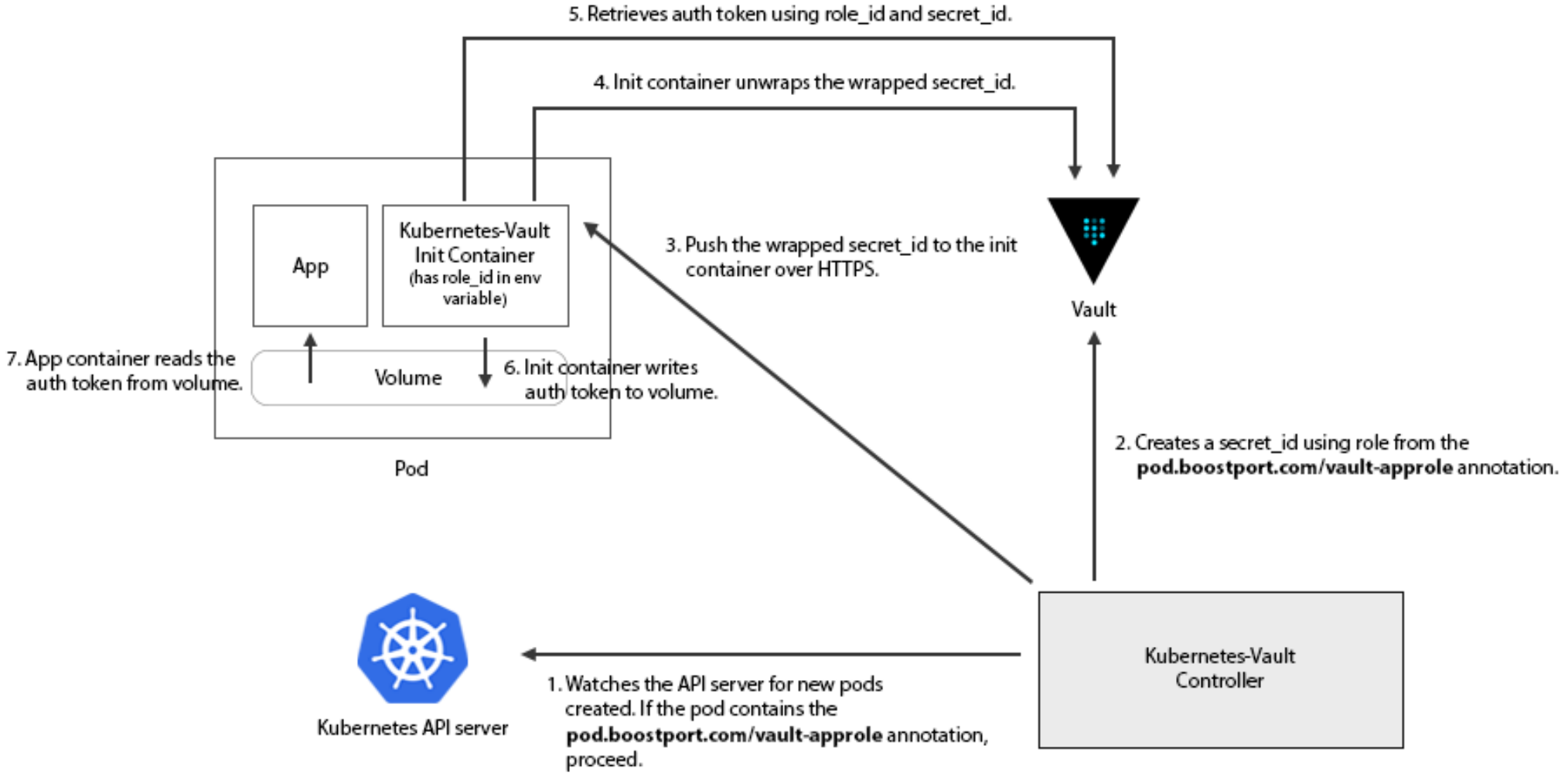
Authorizing service to service calls

- Many customers want to use OAuth 2 for this
- Each service is an OAuth 2 client + resource server
- Option 1:
  - Use client credentials grant
  - Scopes act like permissions
  - AS applies per-client policy to decide which scopes to grant
- Option 2:
  - Service accounts – account has permissions/roles
  - Use ROPC / JWT Bearer grant to authorize clients
  - Limited scope restrictions

How do pods get credentials?

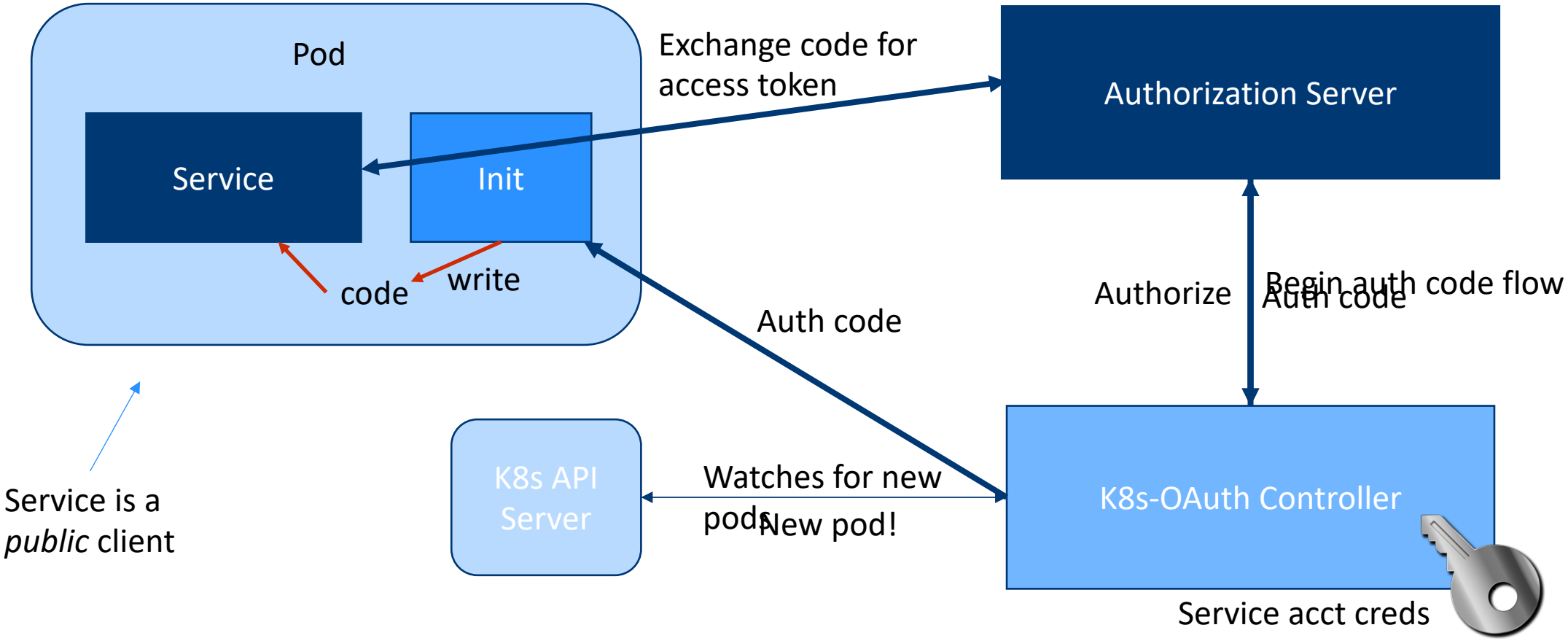
The “Secret Zero” problem

Kubernetes-Vault controller



https://github.com/Boostport/kubernetes-vault

K8s-OAuth controller flow



OAuth mTLS

- draft-ietf-oauth-mtls – (current: draft 13)
- TLS client certificate authentication
  - CA-signed
  - Self-signed
- Certificate-bound access tokens (PoP)
- Access token is bound to client's X.509 cert
- Resource server checks cert matches thumbprint associated with token

4. Public Clients and Certificate Bound Tokens

Mutual TLS OAuth client authentication and certificate-bound access tokens can be used independently of each other. Use of certificate- bound access tokens without mutual TLS OAuth client authentication, for example, is possible in support of **binding access tokens to a TLS client certificate for public clients** (those without authentication credentials associated with the "client_id").

Self-signed mTLS auth code power flow

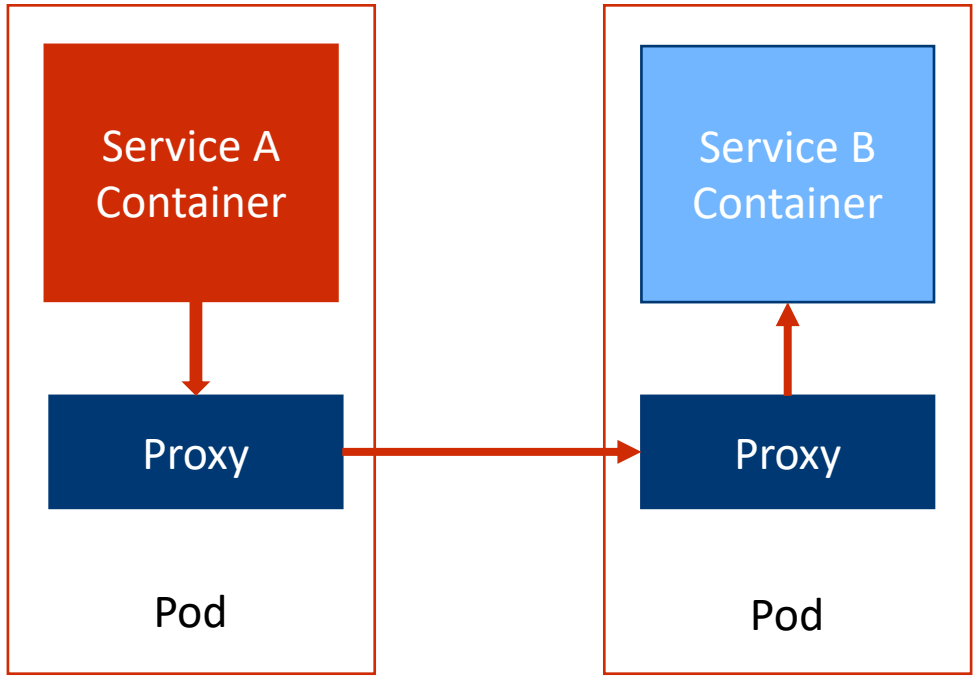
- Pod receives auth code on startup from init container as before
- Application generates self-signed X.509 cert + keypair on startup
- Authenticates connection to Authorization Server during code exchange
- AS ignores cert for authentication (public client)...
- ...but binds access token to the cert

What have we gained?

- Only secret on disk in the clear is a one-time use auth code
  - If already used then attacker gains nothing
  - If attacker wins race then app fails to start and grant is revoked
- Strong mutual TLS-bound access tokens...
- ...without having to run an additional PKI
- Revocation: OAuth token revocation (no CRLs, OCSP)
- Rotation: OAuth refresh token, bind new cert as part of refresh
- Just need normal server certs (e.g., Let's Encrypt)

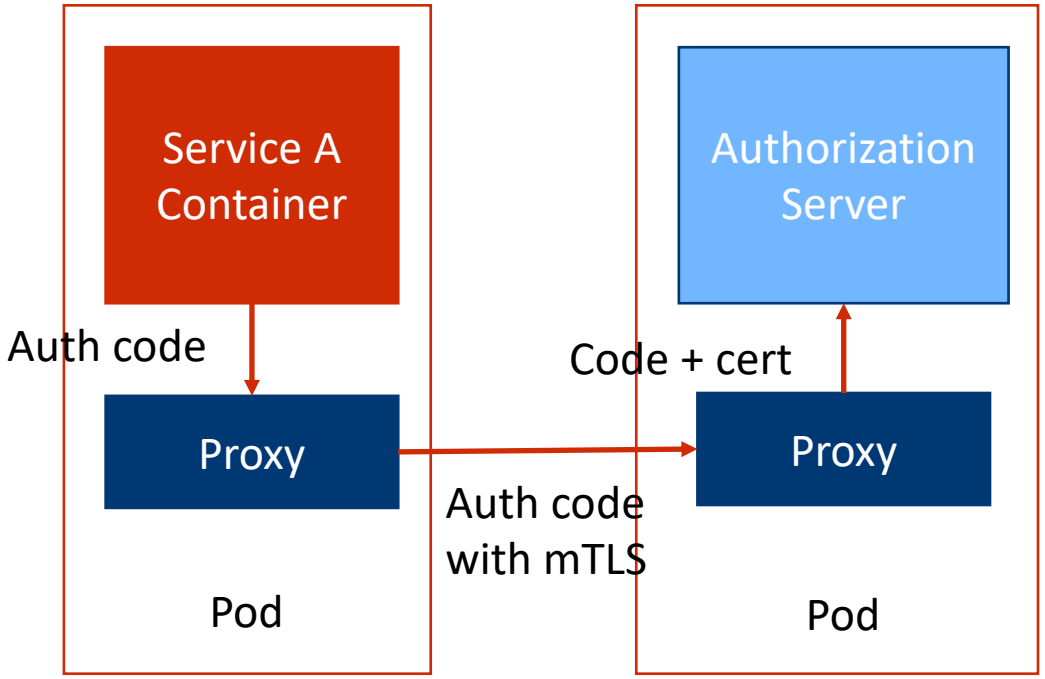
What's a service mesh?

- All communication goes through local proxies
- Deployed as "sidecar" within each pod
- Proxies handle service discovery and routing, load balancing, circuit breaking, metrics, tracing, etc.
- All traffic routed to/from proxy over loopback



Service mesh managed mTLS

- Mesh can transparently add mTLS certificate authentication
- Used for “micro-segmentation” network security
- Can pass-through cert to AS/RS in trusted header
- Access token is bound to cert



FORGEROCK