Deckhouse
Kubernetes Platform

Istio

Federation
Common principles

cluster-a.local

cluster-b.local

Suppose Istio manages two clusters...

cluster-a.local

NS namespace: abc

Pod: foo

Service: foo

cluster-b.local

NS namespace: xyz

Pod: bar

Service: bar

...with applications running in them.

cluster-a.local

Certificates

Root CA

namespace: abc

Pod: foo

Service: foo

cluster-b.local

Certificates

Root CA

namespace: xyz

Pod: bar

Service: bar

Each cluster has a trusted certificate repository that contains a single root certificate of the cluster.

cluster-a.local

Certificates

Root CA

namespace: abc

Pod: foo

Service: foo

cluster-b.local

Certificates

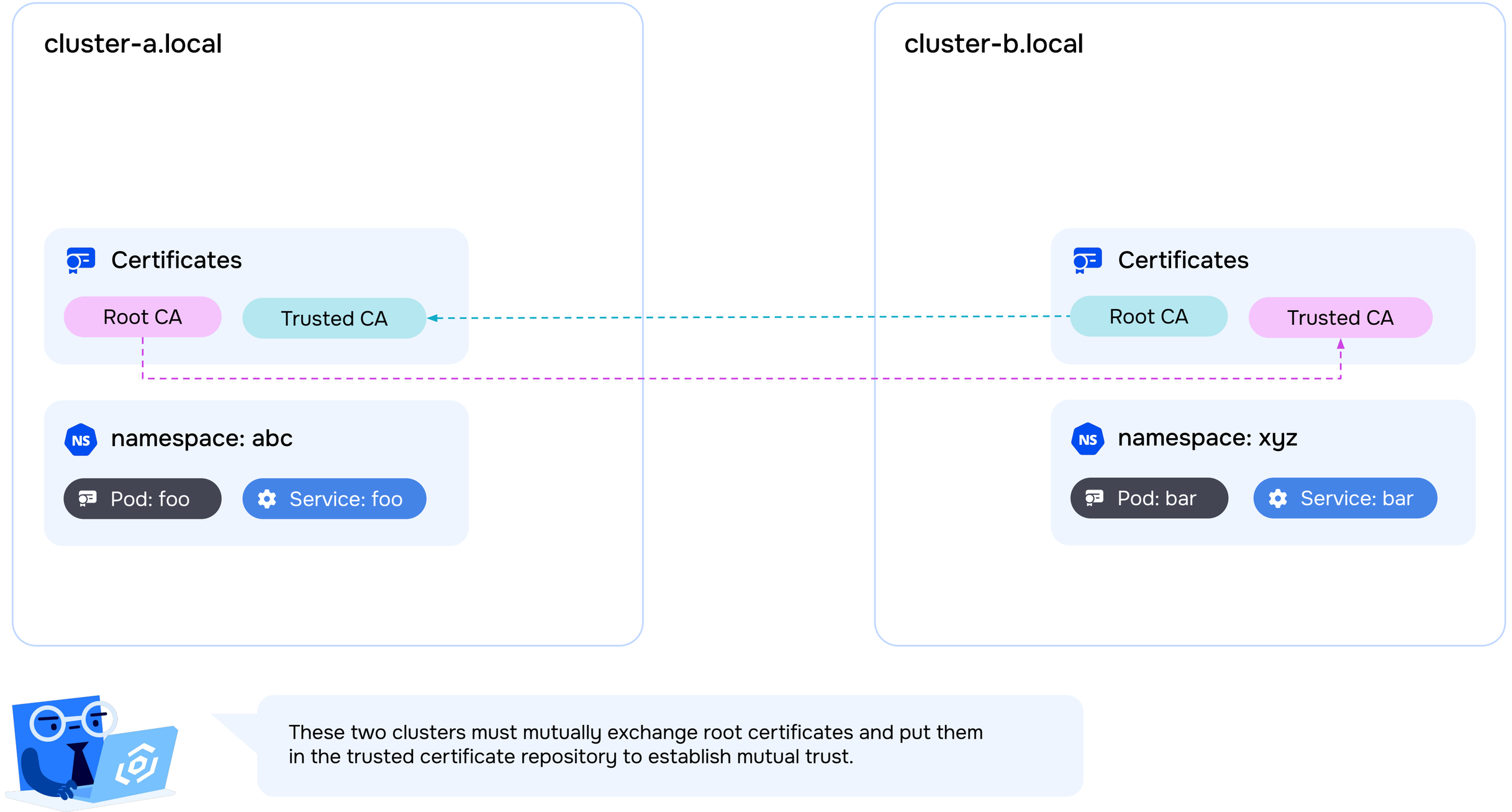
Root CA

namespace: xyz

Pod: bar

Service: bar

These root certificates are used to sign individual Pod certificates for Mutual TLS.



These two clusters must mutually exchange root certificates and put them in the trusted certificate repository to establish mutual trust.

cluster-a.local

Certificates

Root CA

Trusted CA

namespace: abc

Pod: foo

Service: foo

Cloud

Ingress-gateway

Ingress-gateway

cluster-b.local

Certificates

Root CA

Trusted CA

namespace: xyz

Pod: bar

Service: bar

Each cluster has an ingress gateway to receive Mutual TLS requests outside the cluster.

cluster-a.local

Certificates

Root CA

Trusted CA

namespace: abc

Pod: foo

Service: foo

Cloud

Ingress-gateway

Ingress-gateway

cluster-b.local

Certificates

Root CA

Trusted CA

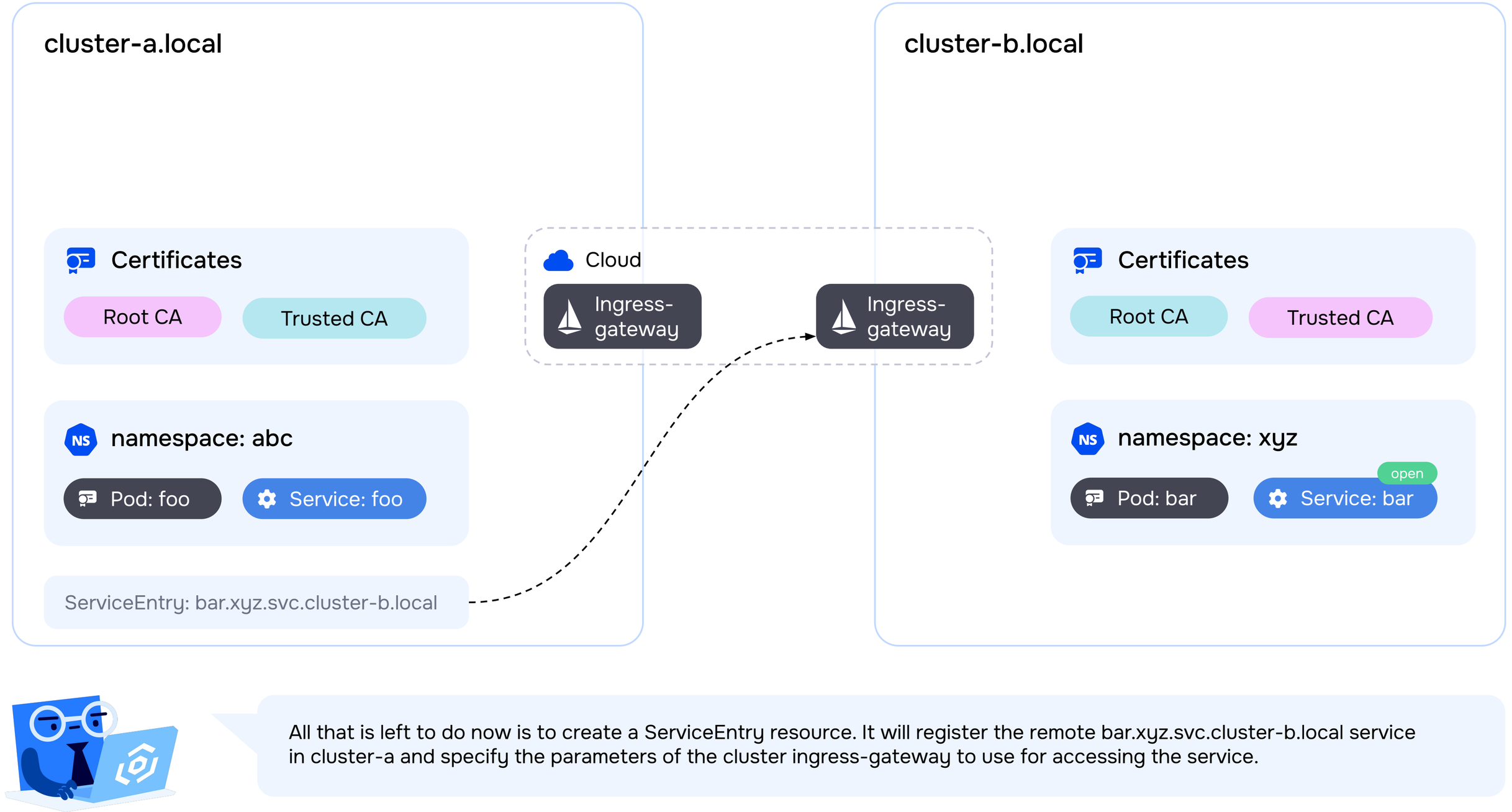
namespace: xyz

Pod: bar

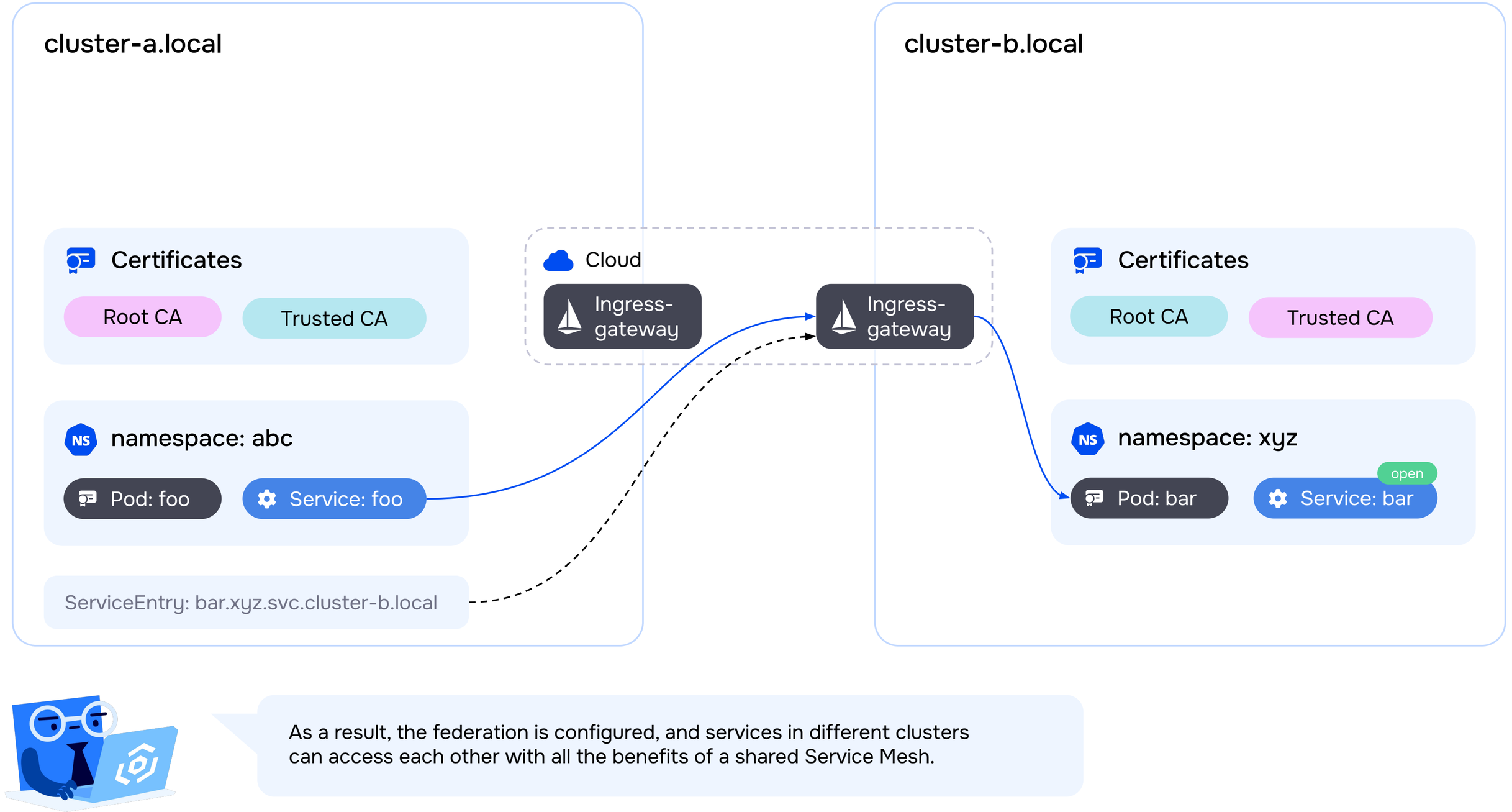
open

Service: bar

We use these ingress gateways to provide external clusters with access to the service under the federation.



All that is left to do now is to create a ServiceEntry resource. It will register the remote bar.xyz.svc.cluster-b.local service in cluster-a and specify the parameters of the cluster ingress-gateway to use for accessing the service.



As a result, the federation is configured, and services in different clusters can access each other with all the benefits of a shared Service Mesh.