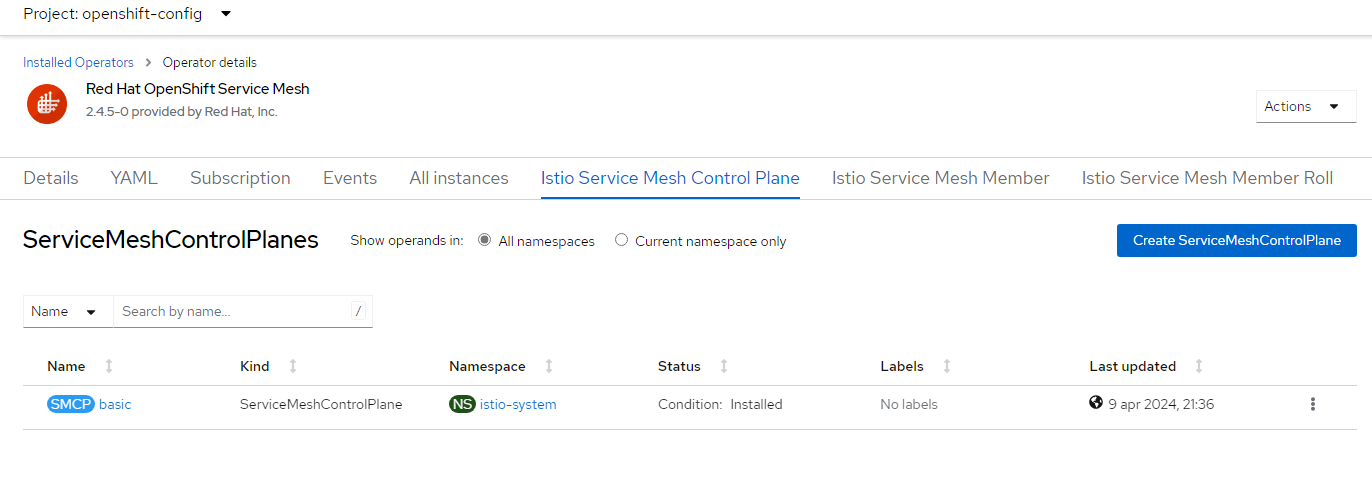
Sotto istio-system creare secret cacerts

Con certificati passati da sicurezza:

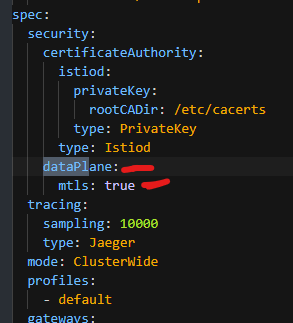
https://docs.openshift.com/container-platform/4.15/service\_mesh/v2x/ossm-security.html#ossm-cert-manage\_ossm-security

oc create secret generic cacerts -n istio-system --from-file=<path>/ca-cert.pem --from-file=<path>/ca-key.pem --from-file=<path>/root-cert.pem --from-file=<path>/cert-chain.pem

da controlPlane poi abilitare mtls



Cliccare su basic 🡪 yaml e aggiungere:



apiVersion: maistra.io/v2

kind: ServiceMeshControlPlane

spec:

security:

dataPlane:

mtls: true

certificateAuthority:

type: Istiod

istiod:

type: PrivateKey

privateKey:

rootCADir: /etc/cacerts

Riavvio pod istio:

oc -n istio-system delete pods -l 'app in (istiod,istio-ingressgateway, istio-egressgateway)'

Verificare che un pod abbia i certificati all’interno:

sleep 60

oc -n bookinfo exec "$(oc -n bookinfo get pod -l app=productpage -o jsonpath={.items..metadata.name})" -c istio-proxy -- openssl s\_client -showcerts -connect details:9080 > bookinfo-proxy-cert.txt

sed -n '/-----BEGIN CERTIFICATE-----/{:start /-----END CERTIFICATE-----/!{N;b start};/.\*/p}' bookinfo-proxy-cert.txt > certs.pem

awk 'BEGIN {counter=0;} /BEGIN CERT/{counter++} { print > "proxy-cert-" counter ".pem"}' < certs.pem

mettere label ai namespace di riferimento:

oc label ns <NAMESPACE> istio-injection=enabled

(maistra.io/member-of=istio-syste

Imola lancia pipeline per injection nei deploy

NB.

Se pod usano hazelcast verificare che anche hazelcast abbia il sidecar istio.

Eventualmente patch a mano per label inject:

oc patch deploy XXX --type merge -p '{"spec": {"template": {"metadata": {"labels": {"sidecar.istio.io/inject": "true"}}}}}'

**A un certo punto qualcuno lancerà le istioPolicy :D**

Parte egress:

cat <<EOF | oc apply -f -

apiVersion: k8s.ovn.org/v1

kind: EgressIP

metadata:

name: eg-3i-scrivaniadigitale-mgw

spec:

egressIPs:

- 10.215.86.91

- 10.215.86.92

namespaceSelector:

matchLabels:

eg-3i-scrivaniadigitale: enable

---

apiVersion: k8s.ovn.org/v1

kind: EgressIP

metadata:

name: eg-3j-schedacliente-mgw

spec:

egressIPs:

- 10.215.86.93

- 10.215.86.94

namespaceSelector:

matchLabels:

eg-3j-schedacliente: enable

---

apiVersion: k8s.ovn.org/v1

kind: EgressIP

metadata:

name: eg-4e-componentitrasversali-mgw

spec:

egressIPs:

- 10.215.86.95

- 10.215.86.96

namespaceSelector:

matchLabels:

eg-4e-componentitrasversali: enable

---

apiVersion: k8s.ovn.org/v1

kind: EgressIP

metadata:

name: eg-ap-anagrafe-mgw

spec:

egressIPs:

- 10.215.86.97

- 10.215.86.98

namespaceSelector:

matchLabels:

eg-ap-anagrafe: enable

EOF

Controllare se i matchLabel dentro egressIP siano popolati

Mettere label al namespace che deve usare egress:

oc label namespace 3i-scrivaniadigitale-mgw eg-3i-scrivaniadigitale=enable

oc label namespace 3j-schedacliente-mgw eg-3j-schedacliente=enable

oc label namespace 4e-componentitrasversali-mgw eg-4e-componentitrasversali=enable

oc label namespace ap-anagrafe-mgw eg-ap-anagrafe=enable

rendere assegnable i nodi che vogliamo utilizzino egress:

oc label node nome-nodo k8s.ovn.org/egress-assignable=""

verificare da un pod per ogni namespace con MSS una wget 10.215.11.33:443

per vedere se ci presentiamo con uno dei due IP censiti nell’egress.