Connect

Red Hat | kyndryl

OpenShift Service Mesh at scale

Steve Mulholland
Senior Solution Architect, Red Hat

Andrew Sewell
Cloud Engineer, Kyndryl

James Force
Principal Consultant, Red Hat

Red Hat | kyndryl

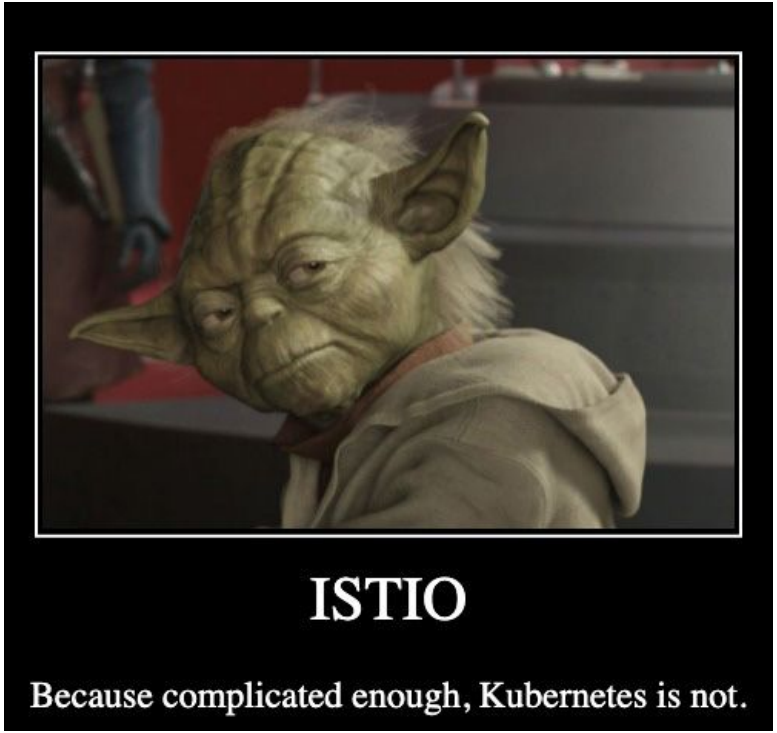
Setting the context.

Security, Operations, Optimisation.

Looking into the future...

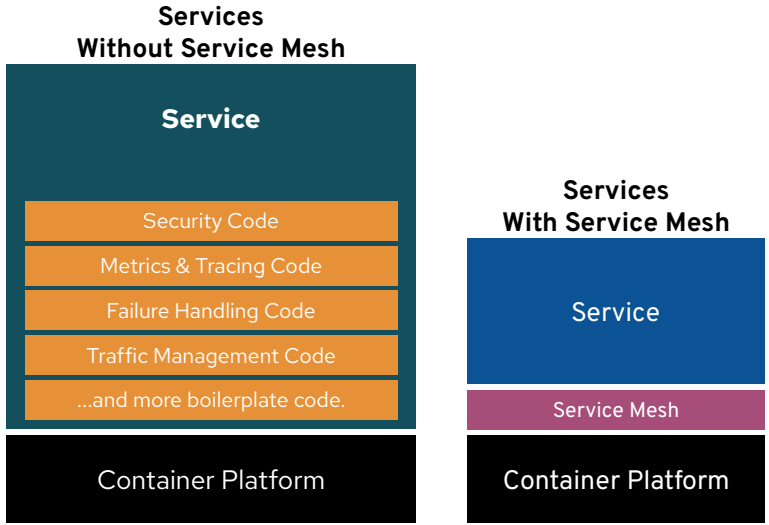
Who is this for?

- Getting started with Service Mesh in small use cases.
- Service Mesh PoC running
- Multi-tenant clusters with no service mesh deployment.



Service Mesh: Who uses it and why?

- **Platform engineers** who aim to make developers happier and more productive by providing:
  - Automatic & enforced mTLS encryption
  - Tools to implement “zero trust” security policies
  - Broad visibility with logs, metrics and traces
  - Network and service failure mitigation
  - Traffic management for migrations and A/B testing
- **Allows developers to focus on business logic**, and not the complexities of microservices.
- There are many ways to achieve these, but a service mesh **“checks all of the boxes”** with one common layer.



OpenShift Service Mesh

OSSM and Istio differences

Community istio

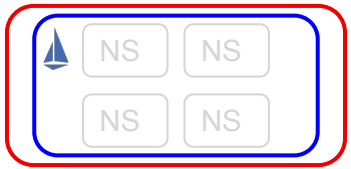
- Helm / istioctl based install
- BoringSSL

OpenShift Service Mesh

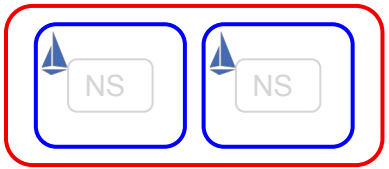
- Operator based install with custom resources to help manage the deployment:
  - Service Mesh Control Plane (SMCP)
  - Service Mesh Member Role (SMMR)
- OpenSSL
- Allows multiple meshes per cluster

Clusters/Mesh Models

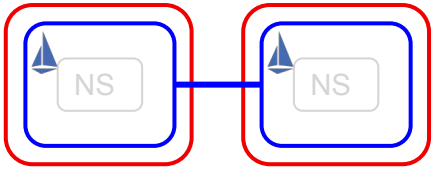
A multi-tenant mesh per cluster



Multiple meshes in a cluster



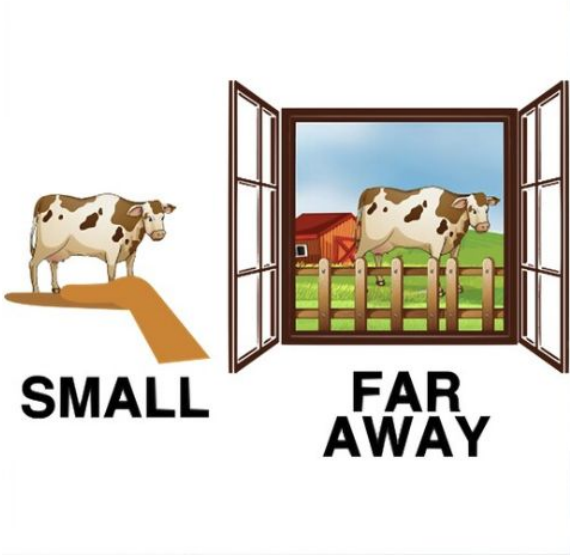
Meshes federated across clusters



- **Clusters**
  - Single Cluster
  - Multiple Federated Custers
  - Multiple non federated clusters
- **Meshes**
  - Single mesh
  - Multiple meshes

Operating at scale

What is scale?



Scale can equally mean large number of small meshes or small number of meshes & clusters with many tenants.

For our purposes scale is:

Openshift Cluster + Single Mesh - Multi-tenant via namespace-based tenant isolation

- 100+ namespaces per cluster
- 1000+ istio proxies per cluster
- Mixture of tenant size / importance / istio understanding

Security

- Isolation
- Hardening
- MTLS

Operations

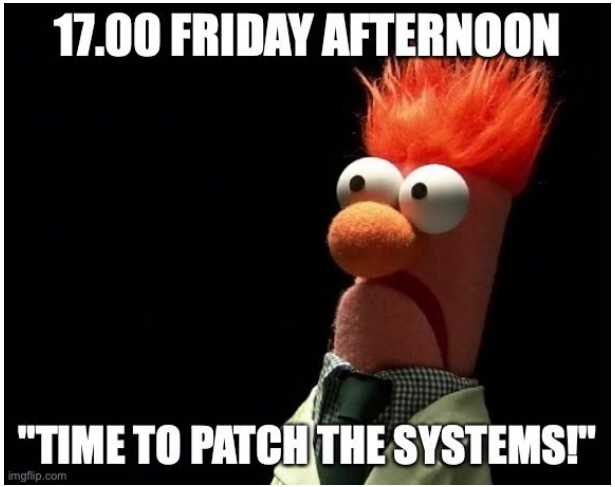
- Patching & upgrades
- Monitoring
- Optimisation

Topologies

- Gateways
- Mesh model

Red Hat | kyndryl

Patching and upgrades



Establish a regular patching regime for Service Mesh

- Patch version - monthly & minor version - quarterly
- Consider tie in with wider Cluster Patching
- Operators - set update approval = manual

Prepare for breaking changes in upgrades (Istio is maturing!)

- Review Service Mesh AND Istio release notes
- Regression testing
- Expect to Triage
    - Forward Fix
    - Application teams have different levels of service mesh expertise

Restarts required! Operators restart the control plane pods not application pods - At scale this requires a degree of coordination for independently restarting app proxies

Monitoring and Observability

Service Mesh v2 ships with dedicated Observability tooling

- Prometheus / Kiali / Grafana / Jaeger
- The nature of the tooling changes with Minor / Major releases
  - v3 – istio control plane decoupled from observability tooling

Prometheus

- Scrapes Envoy + Istiod
  - At scale = 'lots of (too much!) data'
- Mesh Prometheus OOTB is not suited to operating at scale
  - Retention period default is 6 hours
  - No alert manager integration
  - No smcp option to store the Prometheus database on a persistent volume

Monitoring and Observability

Monitoring - Prometheus

Approach depends on the organisation's monitoring / alerting model, however at scale it's worth the effort to 'secure' the Prometheus metrics for longer than 6 hours.

Service Mesh 2.4.x - disable service mesh Prometheus / Grafana and instead scrape via user-workload monitoring.

- Kiali can use this source

Prior to 2.4.x

- create a Service Monitor resource to scrape istiod (envoy) metrics into user-workload monitoring
- Alert manager integration
- kiali still using mesh prometheus

Visualisation - Kiali

By default you can change istio resources in the Kiali UI – this can be disabled via the smcp

```
kiali.dashboard.viewOnlyMode: true
```

At scale it needs additional memory to handle the quantity of metrics.

Tracing - Jaeger

Configure a persistent volume for trace data and set sampling - `spec.tracing.sampling` default is 100%!

Access Logging

Enable globally via `spec.proxy.accesslogging` however, it can't be removed selectively. Locally is possible (envoy filter or Telemetry API)

Optimisations

At scale control plane convergence latency can an issue

- The time it takes for a change in the kube api to be pushed to all proxies

Monitor / alert on pilot_proxy_convergence_time

- Check Pilot_push_triggers to see the source of changes
- Scale out (istiod) or up (cpu / mem limits)

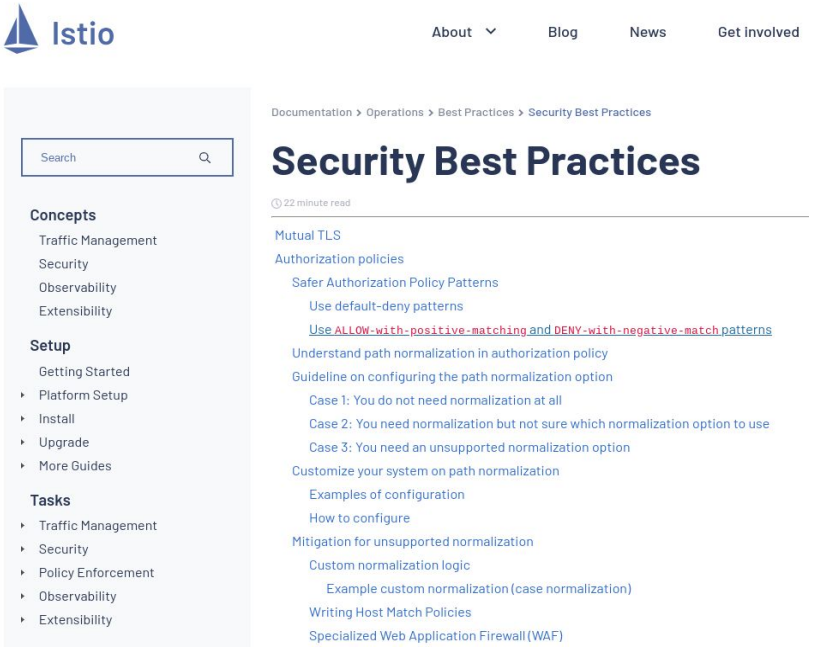
Optimise the size of the istio (envoy) config being pushed

- Create a global sidecar resource to limit cluster config to that which is pertinent to the namespace

Tailor CPU/memory requests & limits

Security

Istio community has a set of security best practices documented



https://istio.io/latest/docs/ops/best-practices/security/

Security

A few things worth shouting out specifically

Enforce mTLS

spec.security.controlPlane.mtls: true

spec.security.dataPlane.mtls: true

spec.security.dataPlane.automtls: true

Destination rules: VERIFY_CERTIFICATE_AT_CLIENT

Enforce Listener TLS ciphers / TLS versions supported

Third Party Access tokens

spec.security.identity.type: ThirdParty

Deny all Network Policies

Built in network policies are ingress only

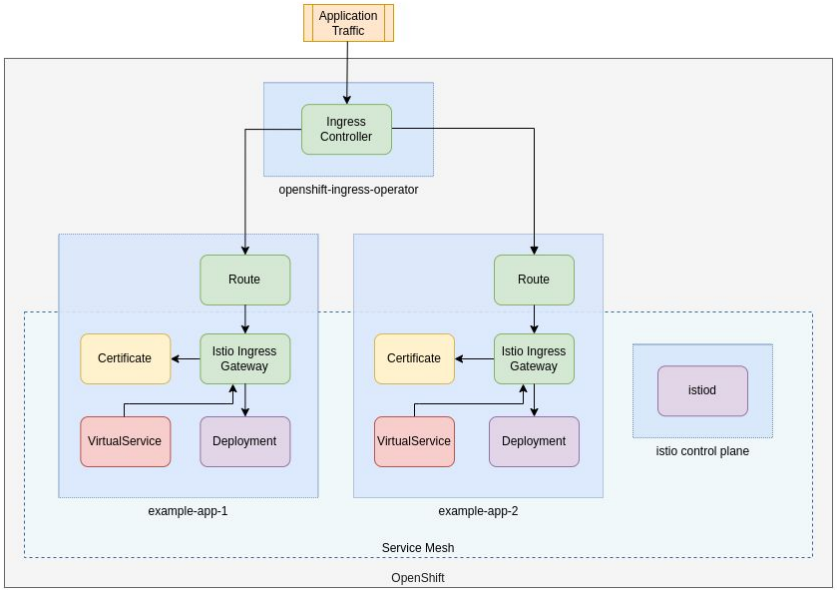
Deny all Authorization Policies

Spec.proxy.networking.trafficControl.outbound.policy:

REGISTRY_ONLY

Topologies

Ingress/Egress Gateways



- Shared Gateways (in control plane namespace) aren't suited to multi-tenant in a single mesh
- Implement gateways in project namespaces
- SMCP has additionalIngressGateway / additionalEgressGateway stanzas
  - Pros – gateway deployments controlled centrally. Gateways patched centrally with control plane
  - Cons – onboarding new projects requires smcp change
- Gateway injection

Topologies

Gateway deployment considerations

“As a security best practice, it is recommended to deploy the gateway in a different namespace from the control plane."

—

istio.io - deploying a gateway

"It may be desired to enforce stricter physical isolation for sensitive services. This can offer a stronger defense-in-depth and help meet certain regulatory compliance guidelines."

—

istio.io - isolate sensitive services

Using auto-injection for gateway deployments is recommended as it gives developers full control over the gateway deployment, while also simplifying operations. This makes the experience of operating a gateway deployment the same as operating sidecars.”

—

Istio.io - deploying a gateway

https://istio.io/latest/docs/setup/additional-setup/gateway/#deploying-a-gateway
https://istio.io/latest/docs/ops/best-practices/security/#isolate-sensitive-services

In conclusion

Operating service mesh at scale you can easily end up herding cats

- Patch regularly
  - Be prepared to triage
- Monitor via Prometheus + Alert via Alert Manager
  - refine / redeploy alerts
- Apply security best practice
  - Be prepared to triage(again!)
  - Polish your TLS handshake troubleshooting skills
    - Tcpdump + envoy debug logging
- Establish clear Ingress / Egress patterns
- Handful of big multi-tenant single mesh clusters versus many single tenant / single mesh clusters

Forward looking into the future

Forward looking into the future

Service Mesh 3

- More of a direct productization of Istio
  - Converge OpenShift Service Mesh with community Istio
  - Support the latest Istio features on OpenShift
  - Increase Red Hat's collaboration with the Istio community
  - Increase cross-platform integrations over customizing Istio
- Based on Istio rather than the forked Maistra project:
  - Maistra CRDs such as the SMCP and SMMR will not be part of Service Mesh 3.
- Continue to use an OpenSSL based Envoy proxy
  - A "bridge-layer" is being contributed upstream to ease maintenance

https://cloud.redhat.com/blog/introducing-a-new-operator-for-istio-on-openshift

Forward looking into the future

Service Mesh 3

- Accelerate support for upstream features e.g.
  - Istioctl
  - Revisions & Canary upgrades of the control plane
  - Multi-cluster topologies such as multi-primary, external control planes
  - Ambient Mesh “sidecar-less” data plane

- Maistra features will be upstreamed, deprecated or moved to separate projects:
  - Multi-tenancy is being implemented in upstream Istio as “Multi-control plane”
  - Istio OpenShift Routing (“IOR”) has been deprecated and will be removed.

https://cloud.redhat.com/blog/introducing-a-new-operator-for-istio-on-openshift

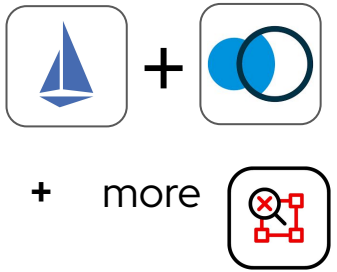
Forward looking into the future

Service Mesh 3

The Service Mesh 3 operator will just manage Istio - Integrations will be managed by separate operators:

OpenShift Service Mesh 2

OpenShift Service Mesh 3



Red Hat Summit

Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat | kyndryl