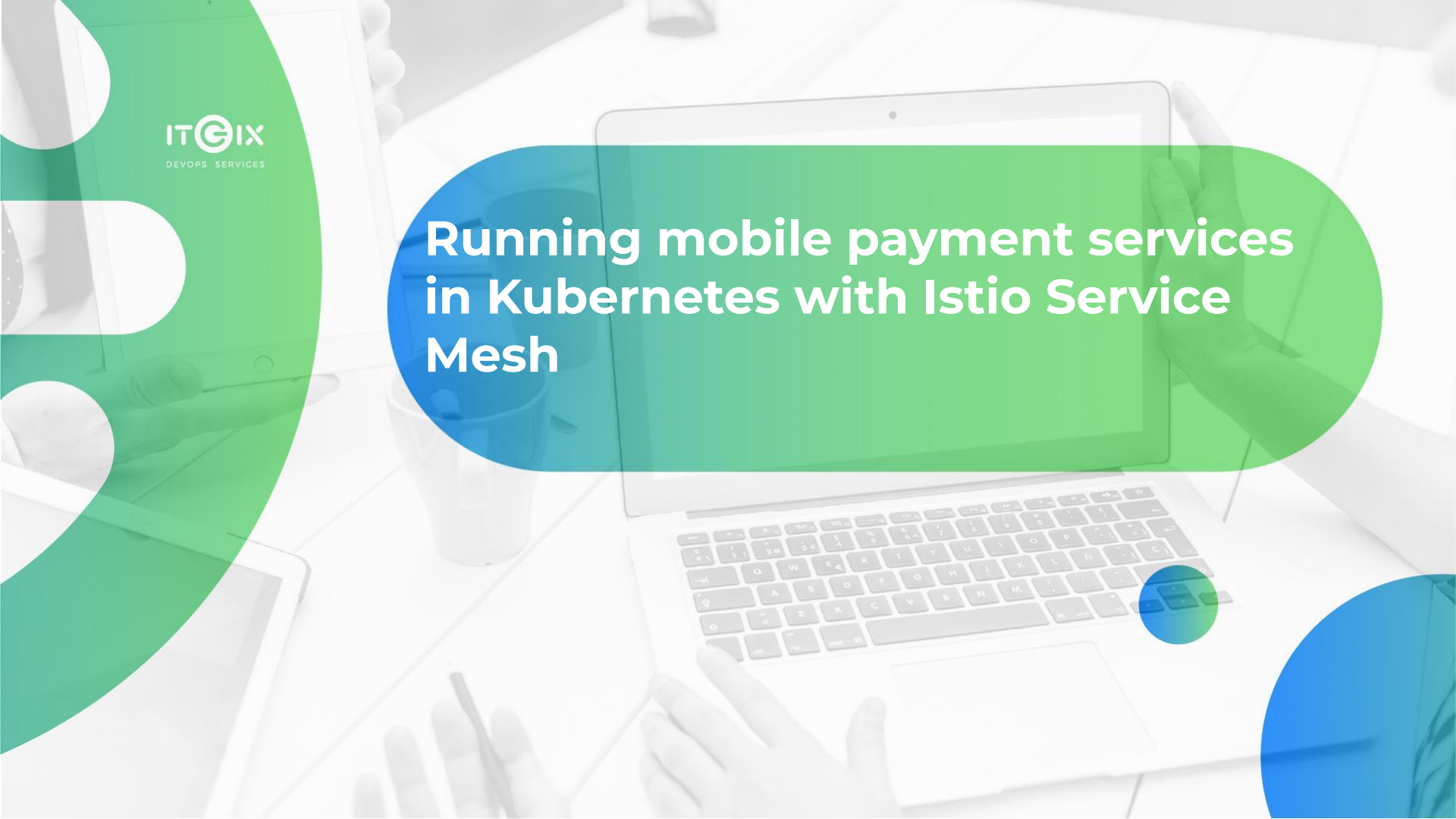
DEVOPS SERVICES



Running mobile payment services in Kubernetes with Istio Service Mesh

Who are we

Daniel is a DevOps engineer with interests in most Open source and Cloud-native tools, and experience in both product, and service-oriented companies.

Mihail is the co-founder of ITGix and CTO for 7 years since the start of the company. About 5 years ago ITGix strategically selected Kubernetes as a main technology to build their service around. Passionate for technologies and always keen to learn new things, working alongside the colleagues in active projects. Outside of work a father of a boy and a girl, likes extreme sports, but primarily snowboarding and fan of motor sports.















#HIRING

Use Case

- Direct carrier billing

Requirements

- Zero Trust
- Integration with many 3rd parties Google APIs and Mobile carriers
- Client certificate authentication with partners
- Reduce MTTR - faster resolution of issues, quick failure pinpointing
- Reduce development time
- Isolation on networking level in cluster
- Whitelist on IPs for partners
- Web Application Firewall for public services with UI

Enter Istio

Istio overview

- First Istio alpha release in 2017
- Joined CNCF 2022
- Sidecar-less architecture in active development
- Utilizes CNCF graduated project Envoy
- Provides Ingress, and Egress Gateways
- Supports both Istio API, and Kubernetes Gateway API, which is in beta
- Multicluster installations

Deployment options

Check the compatibility matrix, and browse the correct documentation version

- Istioctl
- Helm charts
- IstioOperator

We picked the operator

- Generate operator deployment manifests with istioctl
- Dedicated IstioOperator manifests for control plane, ingress/egress, cni
- Persist in git, and overlay with kustomize for different environments
- Reconcile desired state with ArgoCD

IstioOperator resource

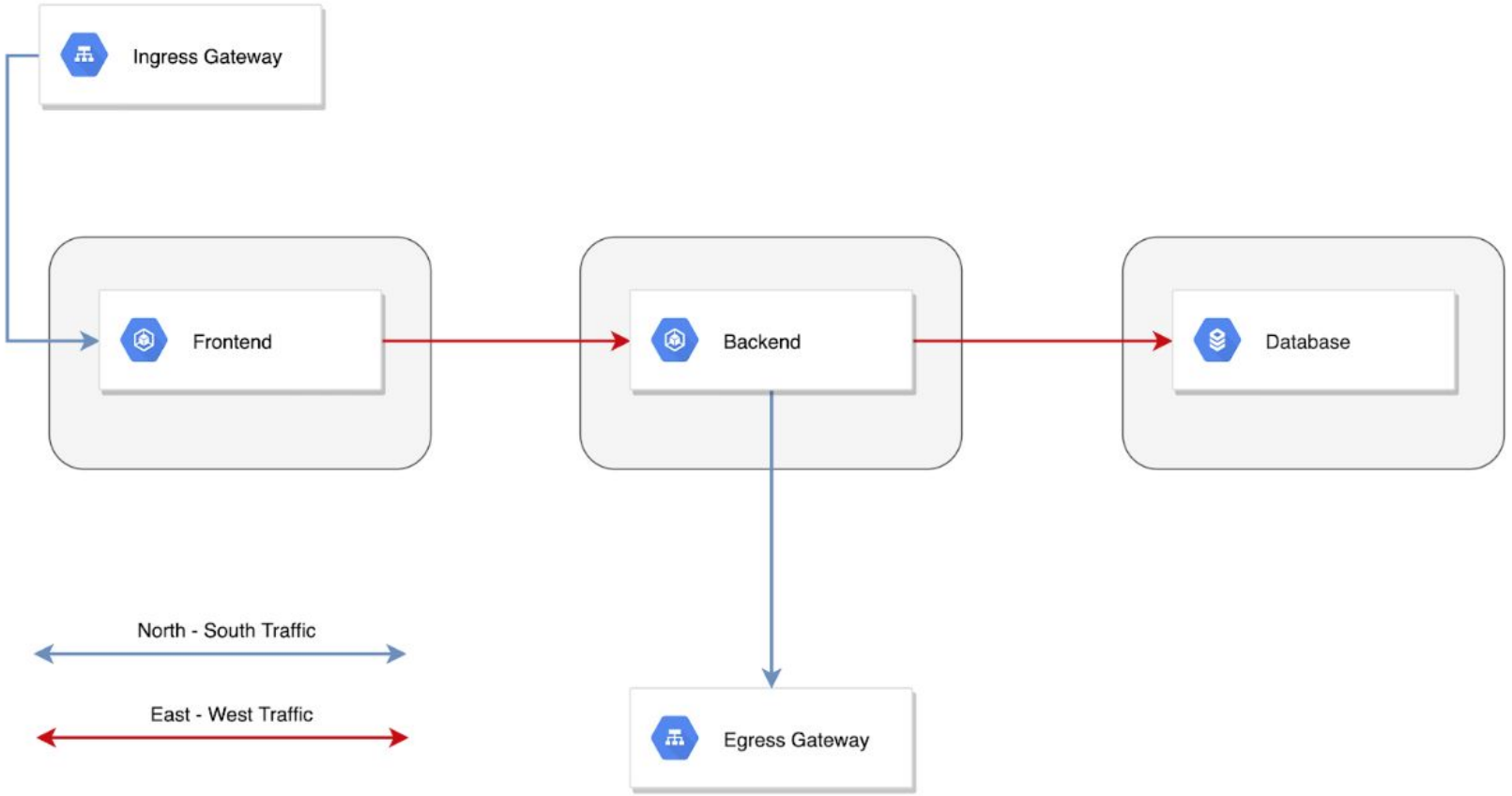
One CRD for all components

- **Istiod**: the Istio control plane. Provides service discovery, configuration and certificate management. Unifies functionality that Pilot, Galley, Citadel and the sidecar injector previously performed, into a single binary. Acts as a Certificate Authority (CA) and generates certificates to allow secure mTLS communication in the data plane.

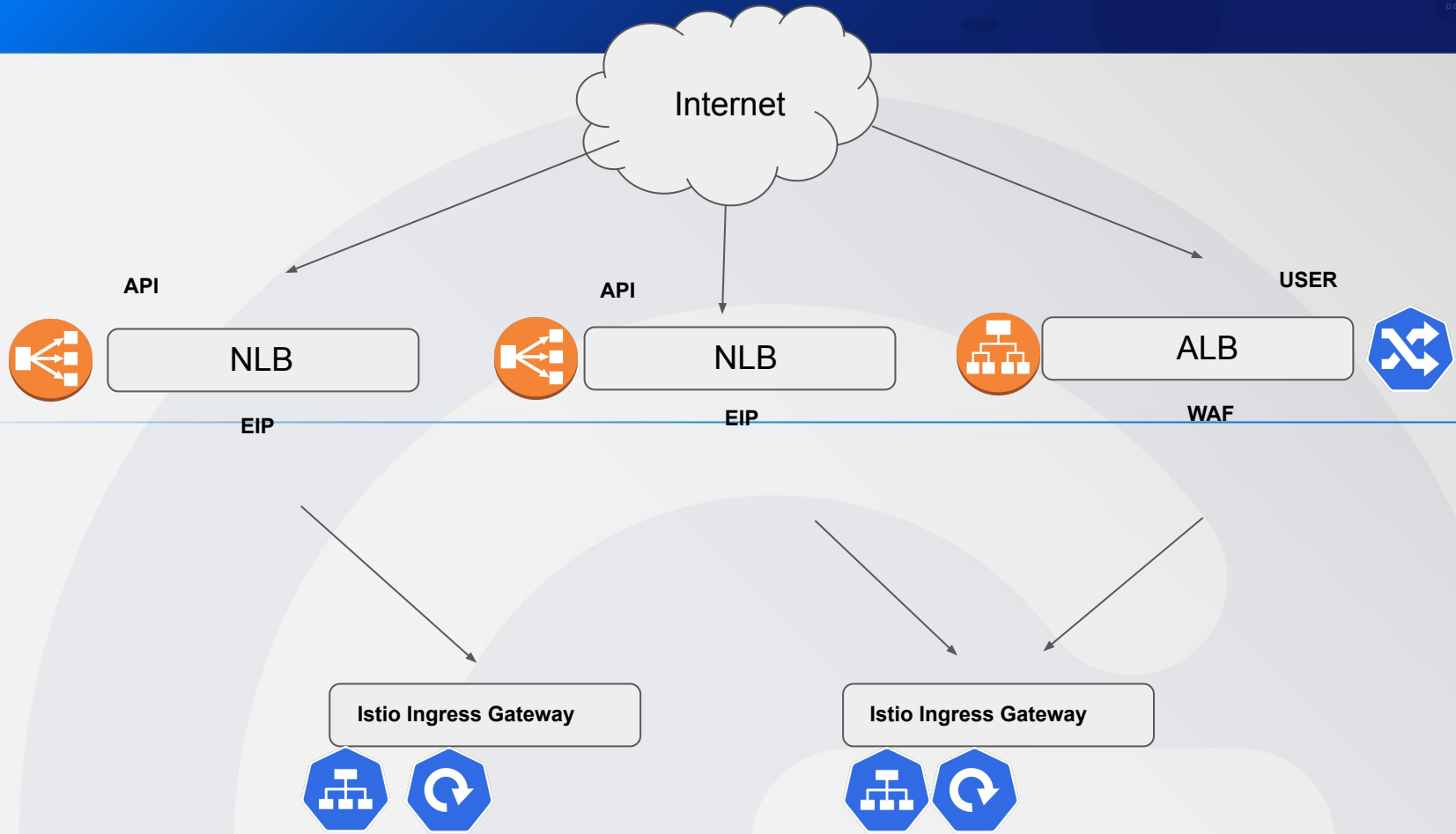
- **Cni**: The Istio CNI plugin identifies user application pods with sidecars requiring traffic redirection and sets this up in the Kubernetes pod lifecycle's network setup phase, thereby removing the requirement for the NET_ADMIN and NET_RAW capabilities for users deploying pods into the Istio mesh. The Istio CNI plugin replaces the functionality provided by the istio-init container.

- **Ingress/Egress Gateways**: Envoy proxies running at the edge of the mesh, providing fine-grained control over traffic entering and leaving the mesh.

Traffic flow



Ingress traffic


Internet
API
NLB
EIP
API
NLB
EIP
USER
ALB
WAF
Istio Ingress Gateway
Istio Ingress Gateway

Configuring Istio Ingress Gateway

- **Gateway** CR: configure gateway listeners, types of protocol, TLS configuration. SNI extension must be included in TLS handshake if gateway is configured with “Host”.

- **VirtualService** CR: defines a set of application-layer traffic routing rules to apply when host is addressed. Routing to subsets (requires **DestinationRule** CR), route, and headers manipulation, timeouts, retries.

- **AuthorizationPolicy** CR: CUSTOM, DENY and ALLOW actions for access control. Selects sources based on principals, namespaces, IPs. Selects operations based on host, method, ports, paths.

Resource examples

```
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: my-gateway
  namespace: some-config-namespace
spec:
  selector:
     app: ingress-gateway
  servers:
  - port:
     number: 443
     name: https
     protocol: HTTPS
     hosts:
     - eu.bookinfo.com
     tls:
     mode: MUTUAL
     serverCertificate:
/etc/certs/servercert.pem
     privateKey: /etc/certs/privatekey.pem
     caCertificates: /etc/certs/CA.crt
     minProtocolVersion: TLSV1_2
```

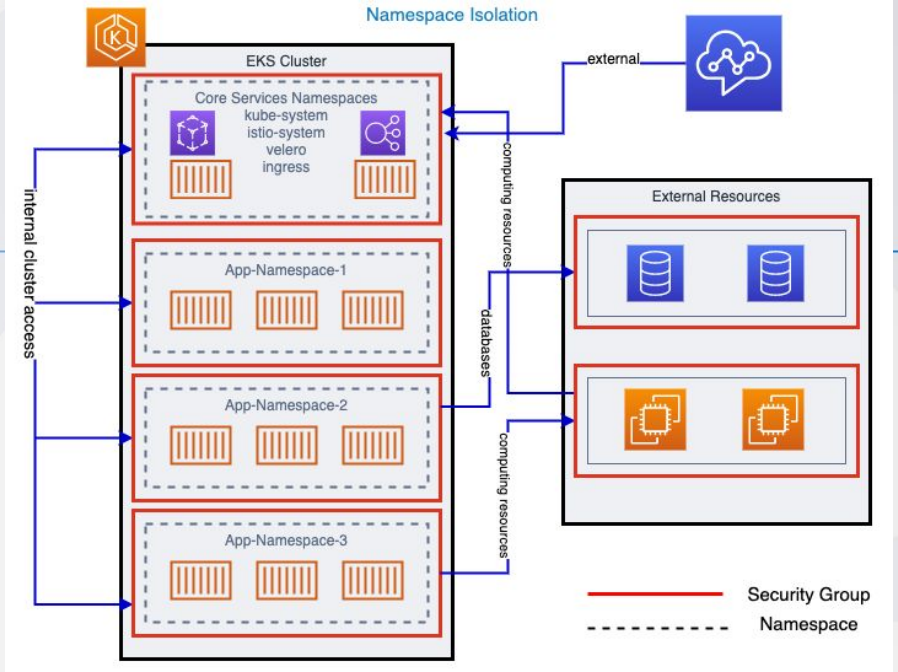
```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: bookinfo-rule
  namespace: bookinfo-namespace
spec:
  hosts:
  - eu.bookinfo.com
  gateways:
  - some-config-namespace/my-gateway
  http:
  - match:
    - headers:
        cookie:
          exact: "user=dev-123"
    route:
    - destination:
        port:
          number: 7777
        host: reviews.qa.svc.cluster.local
```

East-west traffic management

Zero Trust , mlts, net pols

Zero Trust policy

- Namespace isolation



Me configuring the network policies

Automate with helm

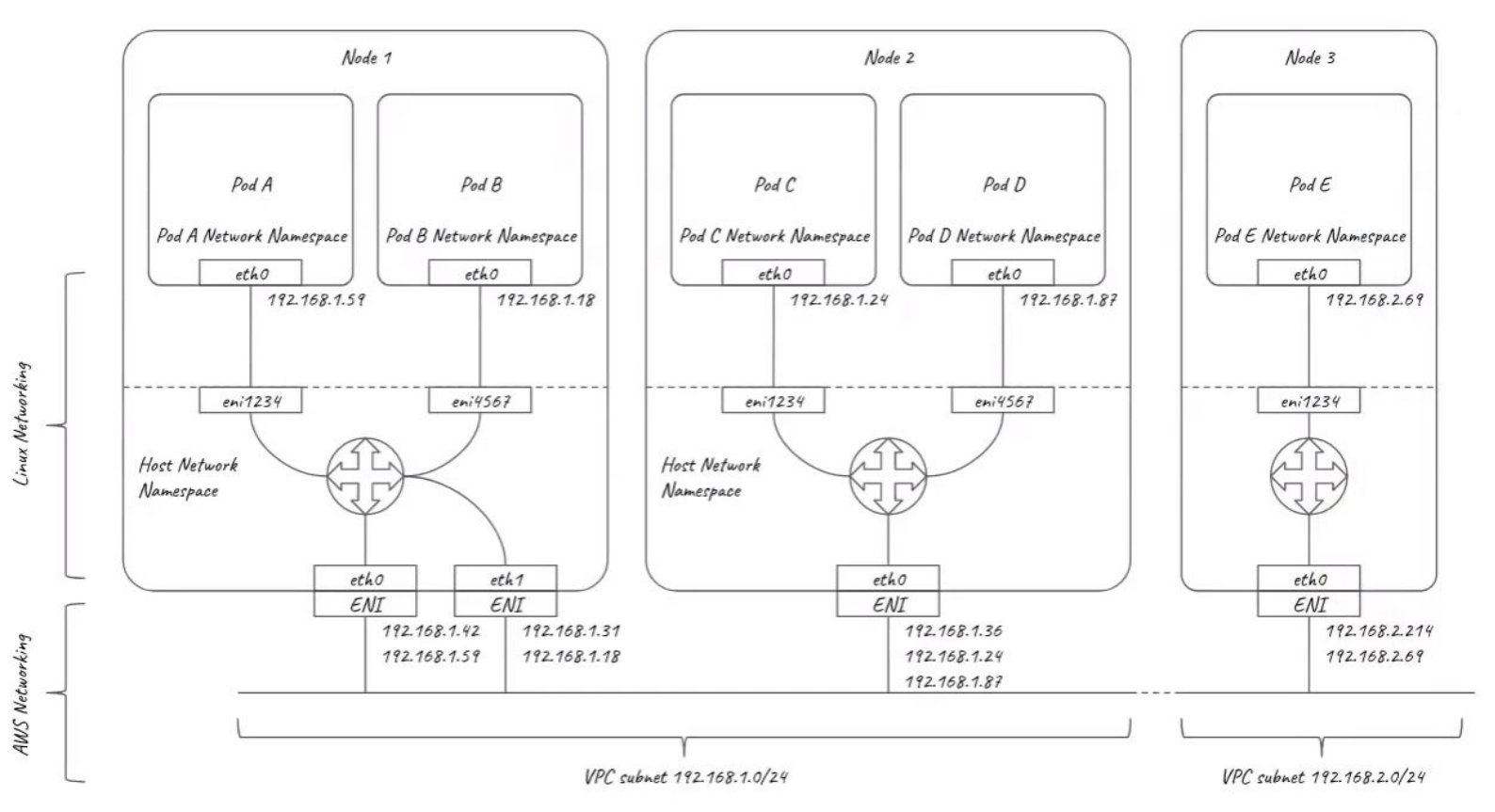
Helm template

```
{{- if .Values.global.networkpolicy.enabled -}}
{{- if .Values.global.istioEgressGateway.enabled }}
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: istio-automatic-gw-netpol
spec:
  egress:
 {{- range .Values.global.istioIngressGateway.ingressEndpoints }}
 {{- $ez := . }}
{{- range .serviceNamespace }}
    - to:
        - namespaceSelector:
            matchLabels:
              app.kubernetes.io/name: {{.}}
      ports:
        - port: {{$ez.internalport}}
          protocol: TCP
{{- end }}
{{- end }}
```

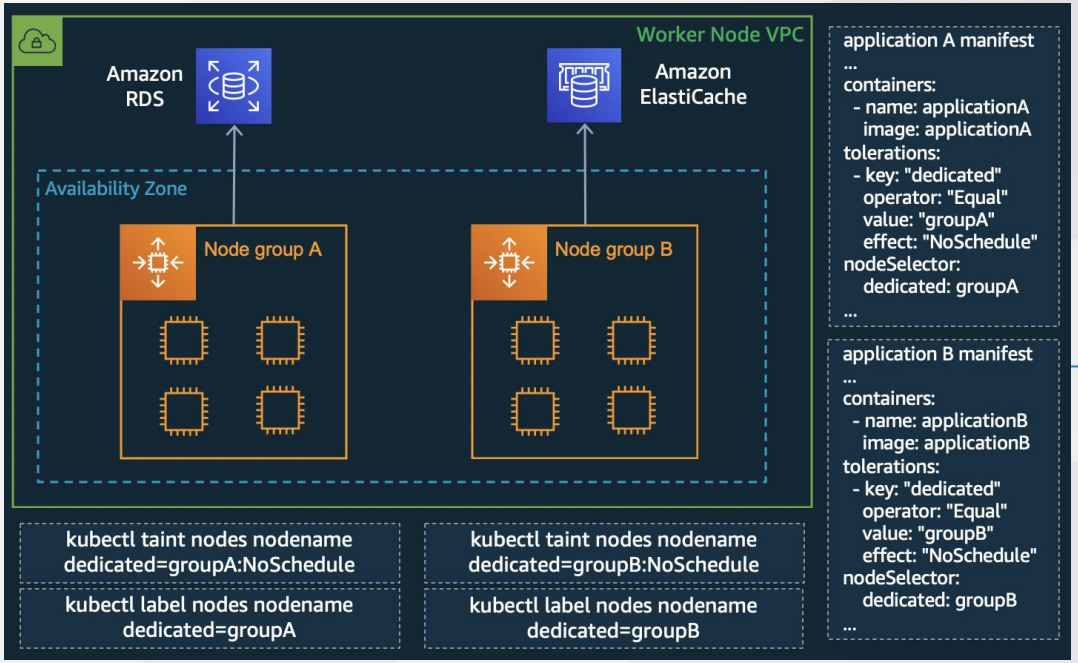
Resulting network policy resource

```
spec:
  egress:
  - ports:
    - port: 8540
      protocol: TCP
    to:
    - namespaceSelector:
        matchLabels:
          app.kubernetes.io/name: app-namespace1
  - ports:
    - port: 8870
      protocol: TCP
    to:
    - namespaceSelector:
        matchLabels:
          app.kubernetes.io/name: app-namespace2
  - ports:
    - port: 8670
      protocol: TCP
    To:

…

<100 more>
```

AWS limited network policies



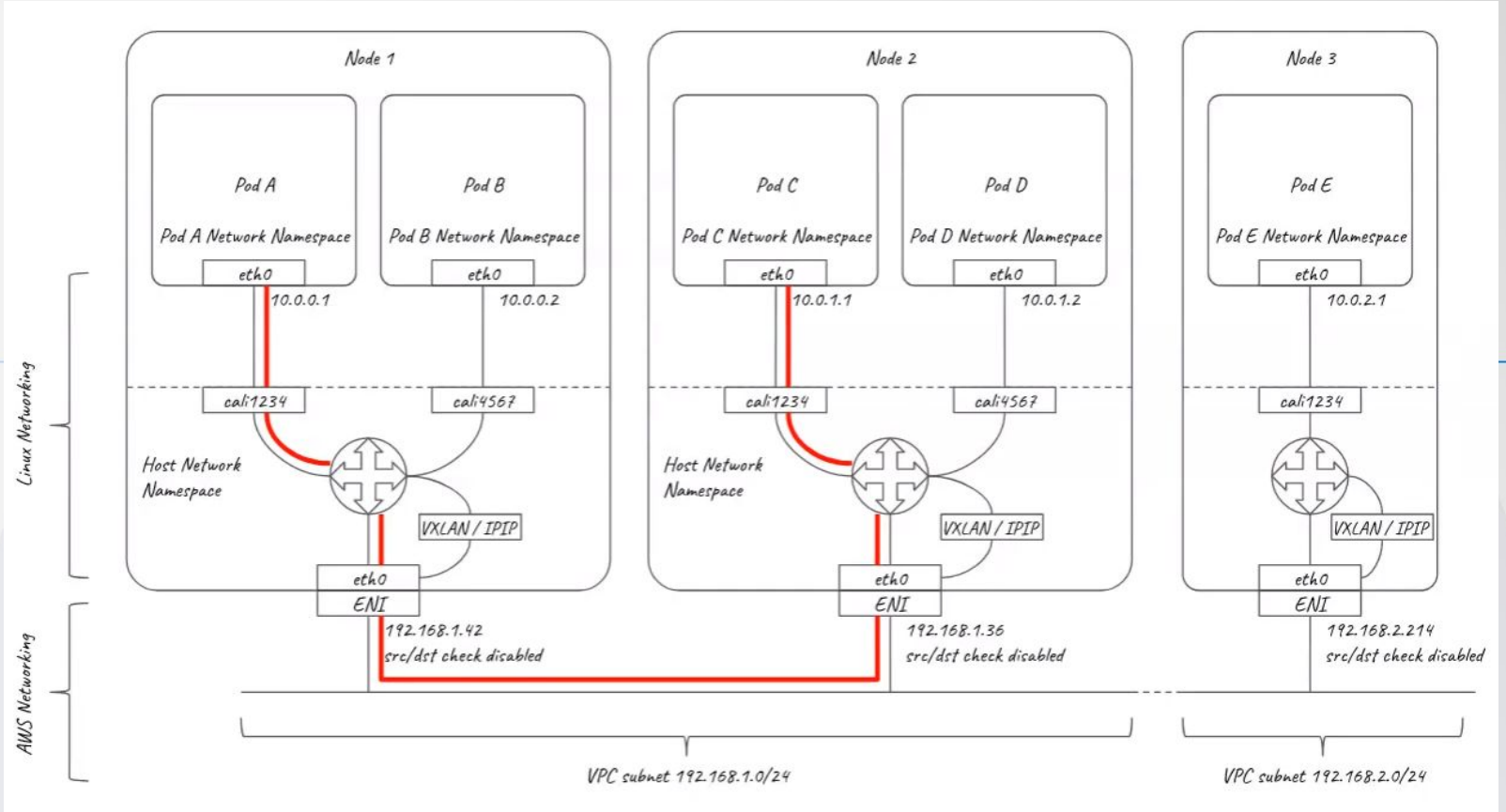
Security groups for Pods




```
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: <name-of-sg-policy>
  namespace: <namespace>
spec:
  podSelector: {}
  securityGroups:
    groupIds:
      - <app specific security-group-id>
```

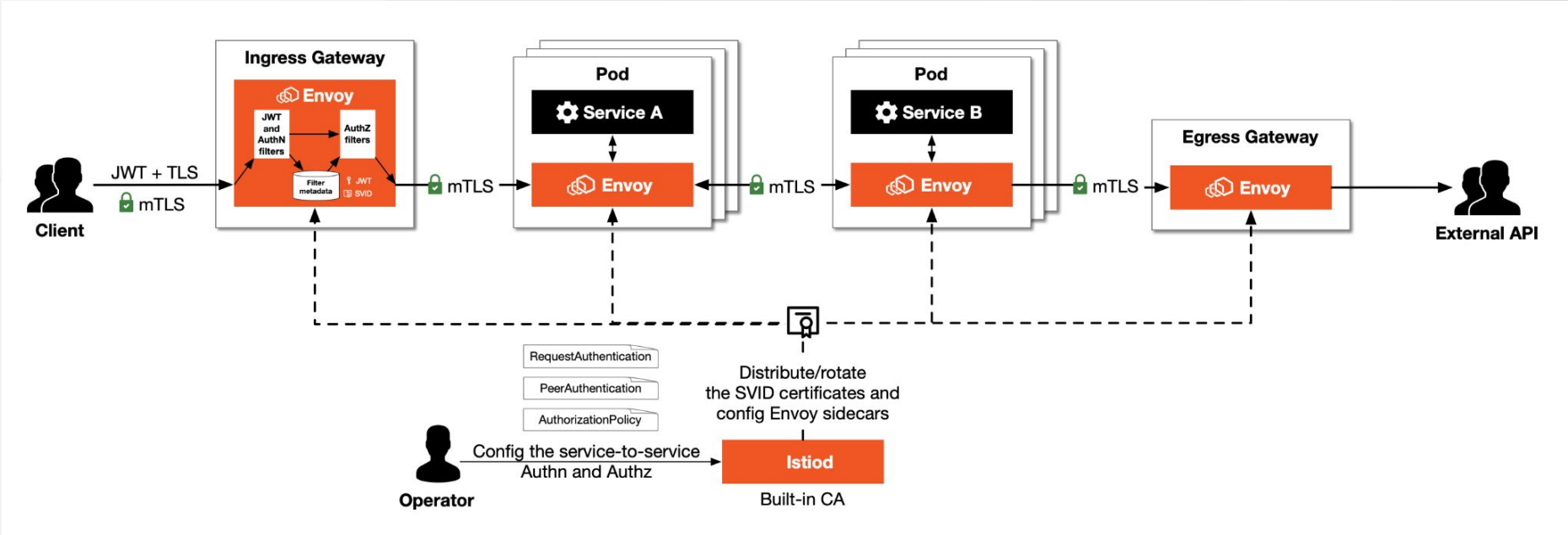
```
Example:
NAME                          SECURITY-GROUP-IDS

netpol-default-sg          [sg-0d8e73xxxxxx2]
netpol-custom-sg           [sg-0bfaxxxxxx649]
```

Calico network policies



Securing EAST - WEST traffic (service-to-service communication)

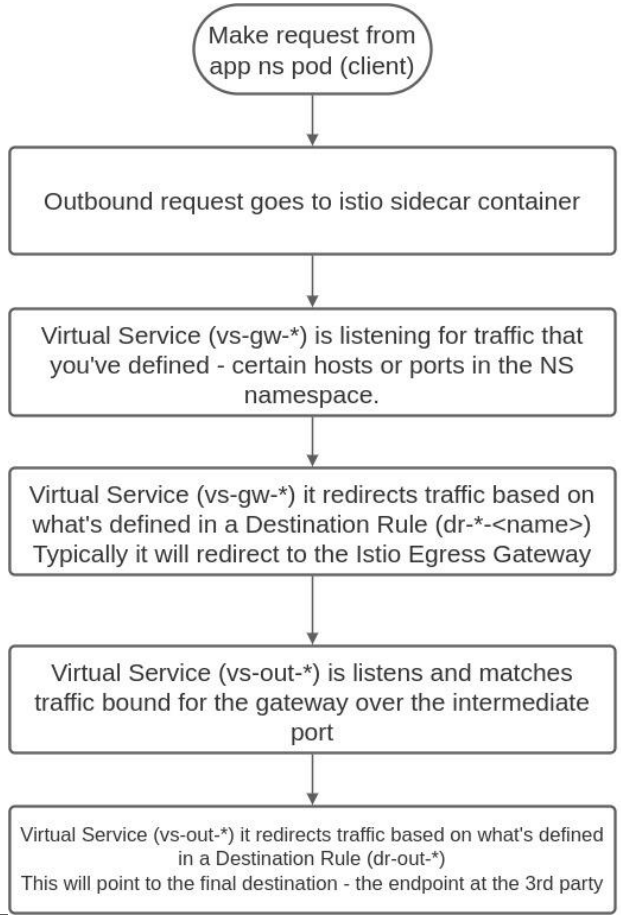
mTLS in-cluster communication



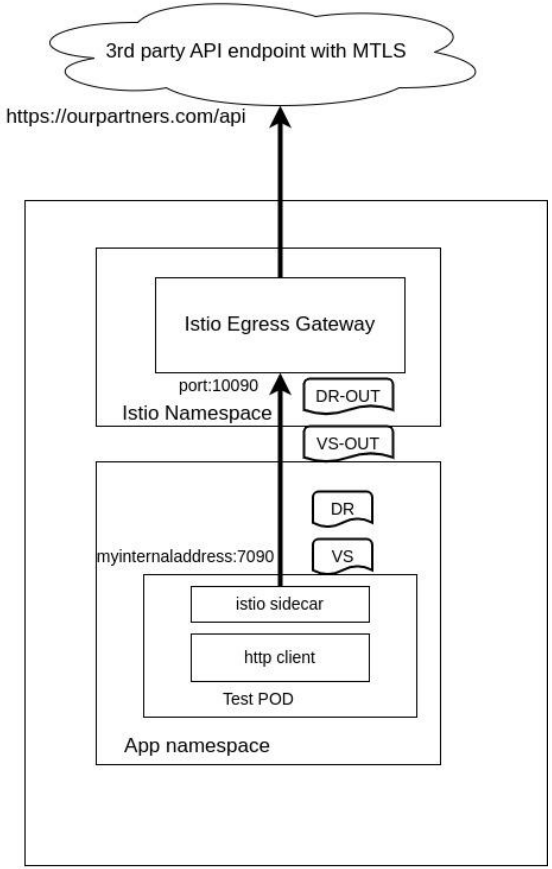
Egress traffic

Connect to 3rd parties securely

Egress gateway capture of requests



Virtual Service is defined in the Gateway NS, but the exportTo copies it to the other NS



Examples

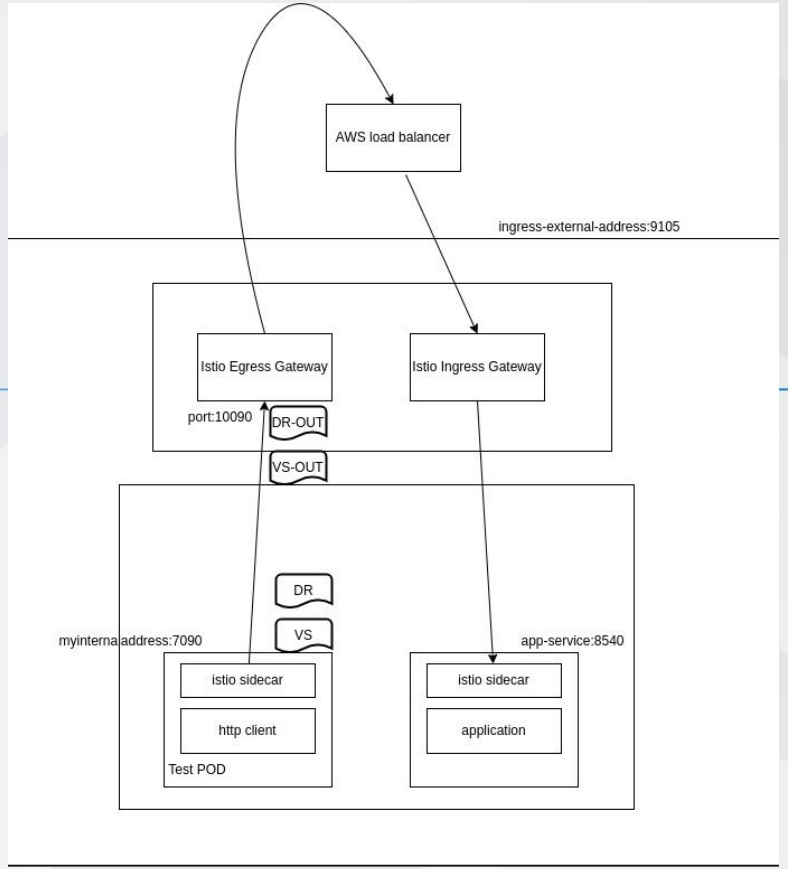
```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
...
spec:
  exportTo:
  - .
  - app-namespace1
  - app-namespace2
  gateways:
  - mesh
  hosts:
  - '*'
  http:
  - match:
    - gateways:
      - mesh
      headers:
        host:
          prefix: myinternaladdress
      port: 7090
      uri:
        prefix: /i1/
    rewrite:
      authority: ourpartners.com
    route:
    - destination:
        host: egress-gateway-external.istio-gateways.svc.cluster.local
        port:
          number: 10090
        subset: dr-ourpartners-proxyout
```

```yaml
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
...
spec:
  exportTo:
  - .
  - app-namespace1
  - app-namespace2
  gateways:
  - istio-gateways/egress-gateway-external
  hosts:
  - ourparthers.com
  http:
  - match:
    - gateways:
      - istio-gateways/egress-gateway-external
      port: 10090
      uri:
        prefix: /i1/
    rewrite:
      authority: ourparthers.com
      uri: /
    route:
    - destination:
        host: ourpartners.com
        port:
          number: 443
        subset: dr-out-ourparthers-proxyout
      weight: 100
```

Examples

```yaml
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: dr-out-ourparthers-proxyout
spec:
  exportTo:
  - .
  - app--str--istio-gateways
  host: ourpartners.com
  subsets:
  - name: dr-out-ourparthers-proxyout
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
      portLevelSettings:
      - port:
          number: 443
        tls:
          clientCertificate: /etc/istio/egressgateway-certs/self-signed-itgix-client-2022.crt
          mode: MUTUAL
          privateKey: /etc/istio/egressgateway-certs/self-signed-itgix-client-2022.key
          sni: ourpartners.com
```

Hairpin testing



Day 2 - Upgrading

Canary upgrades with Istio operator

- New version of istiod is deployed side by side
- Ingress/Egress gateways are upgraded in-place
- New revision is explicitly set in desired namespace annotation
- Sidecar proxies need workload rollouts to pick up on the revision change
- Run jmeter probes every step of the way
- No downtime observed
- Only minor increase in latency, until istiod pushes configuration to new gateways
- Upgrading across up to two minor versions should be OK

Day 2 - Monitoring

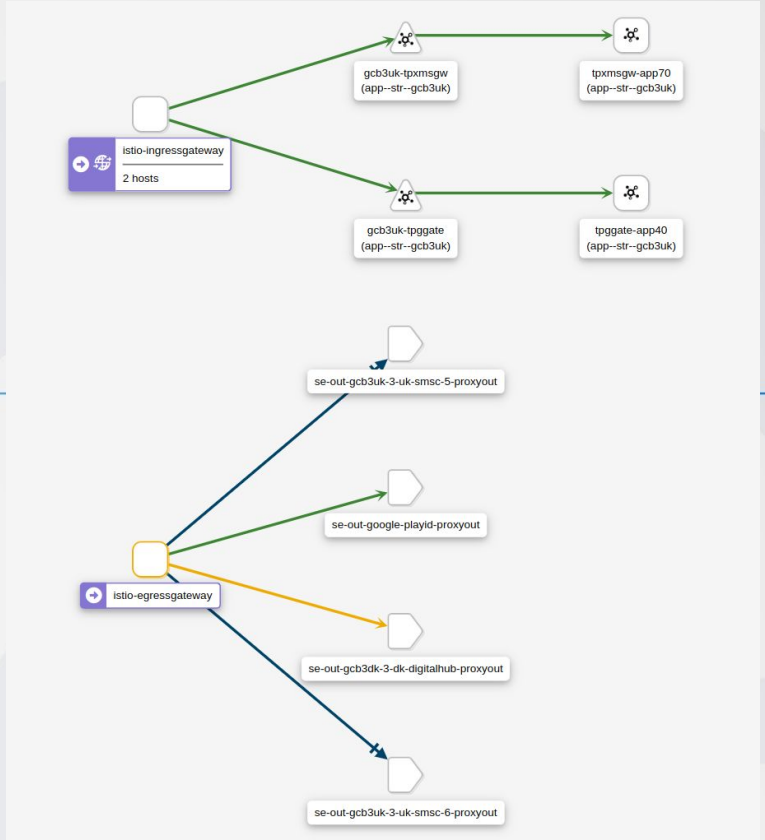
Envoy metrics and access logs

- Control plane, poxy-level, and service-level metrics
- Dedicated Prometheus instance with federation or Thanos sidecar enabled
- Exported metrics, and labels may be customized
- Source information from Envoy request / response/ connection attributes
- Access logs may be enabled or disabled per workload
- Access log format is configurable
- Envoy terminology

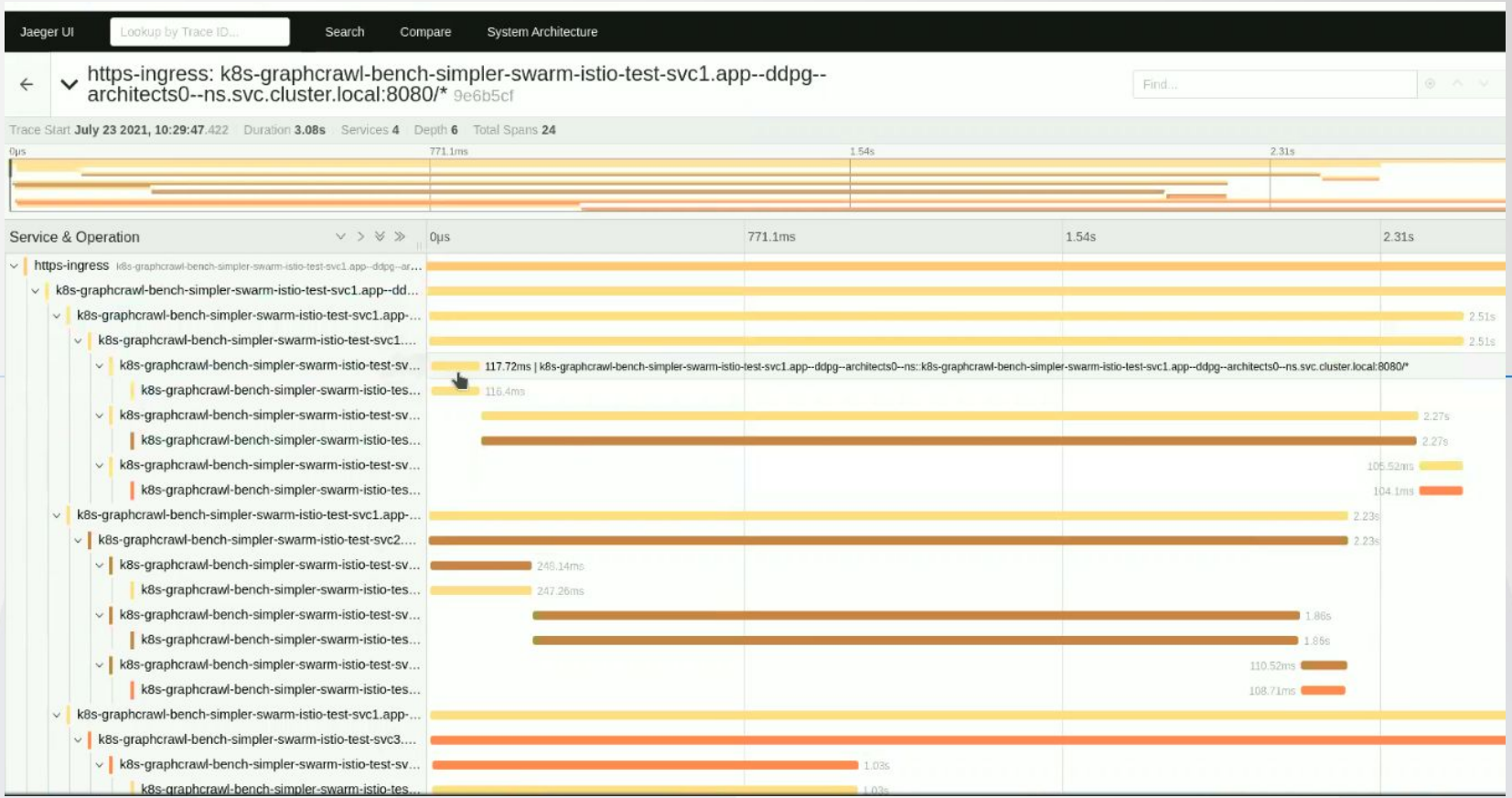
Istio Prometheus metrics

Communication visualization

- Kiali
- SysDig



Distributed tracing - Jaeger



Summary

Requirements:

- Zero Trust
  - mtls
- Client certificate authentication with partners
  - Ingress and Egress gateway and destination rules
- Reduce MTTR - faster resolution of issues, quick failure pinpointing
  - Prometheus , Jaeger, Kiali
- Reduce development time
  - move client cert auth. to Istio
- Isolation on networking level in cluster
  - network policies
- Whitelist on IPs for partners
  - Authorization policies

Thank you !!!
Q&A

@ mihail.vukadinoff@itgix.com
@ daniel.milanov@itgix.com
https://www.linkedin.com/in/mihail-vukadinoff-74371155/

Useful links

References and used sources

https://istio.io/
https://docs.aws.amazon.com/eks/latest/userguide/calico.html
https://www.youtube.com/watch?v=J1VbZR7j4sI&t