The bridge to possible

Explore complexities and best practices for deploying applications in multi cluster service mesh

Sundar Srinivasaraghavan – Principal Architect
Ravi Jandyala – Product Management Architect

BRKCLD-2019

Cisco Webex App

https://ciscolive.ciscoevents.com/ciscolivebot/#BRKCLD-2019

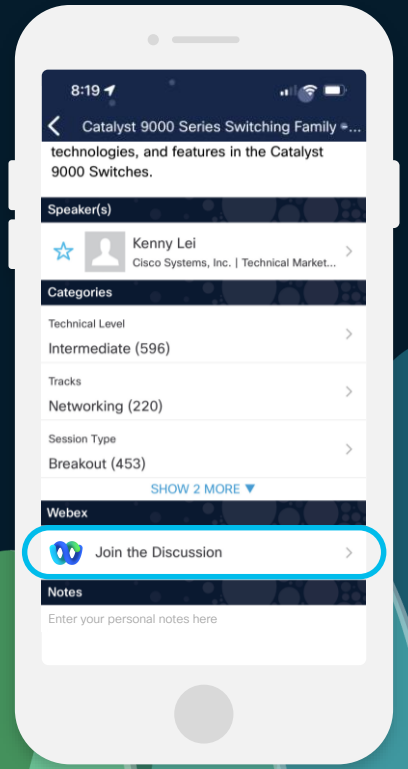
Questions?

Use Cisco Webex App to chat with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 7, 2024.

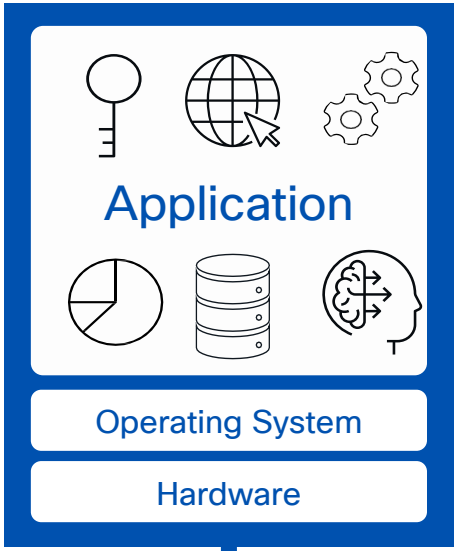


Agenda

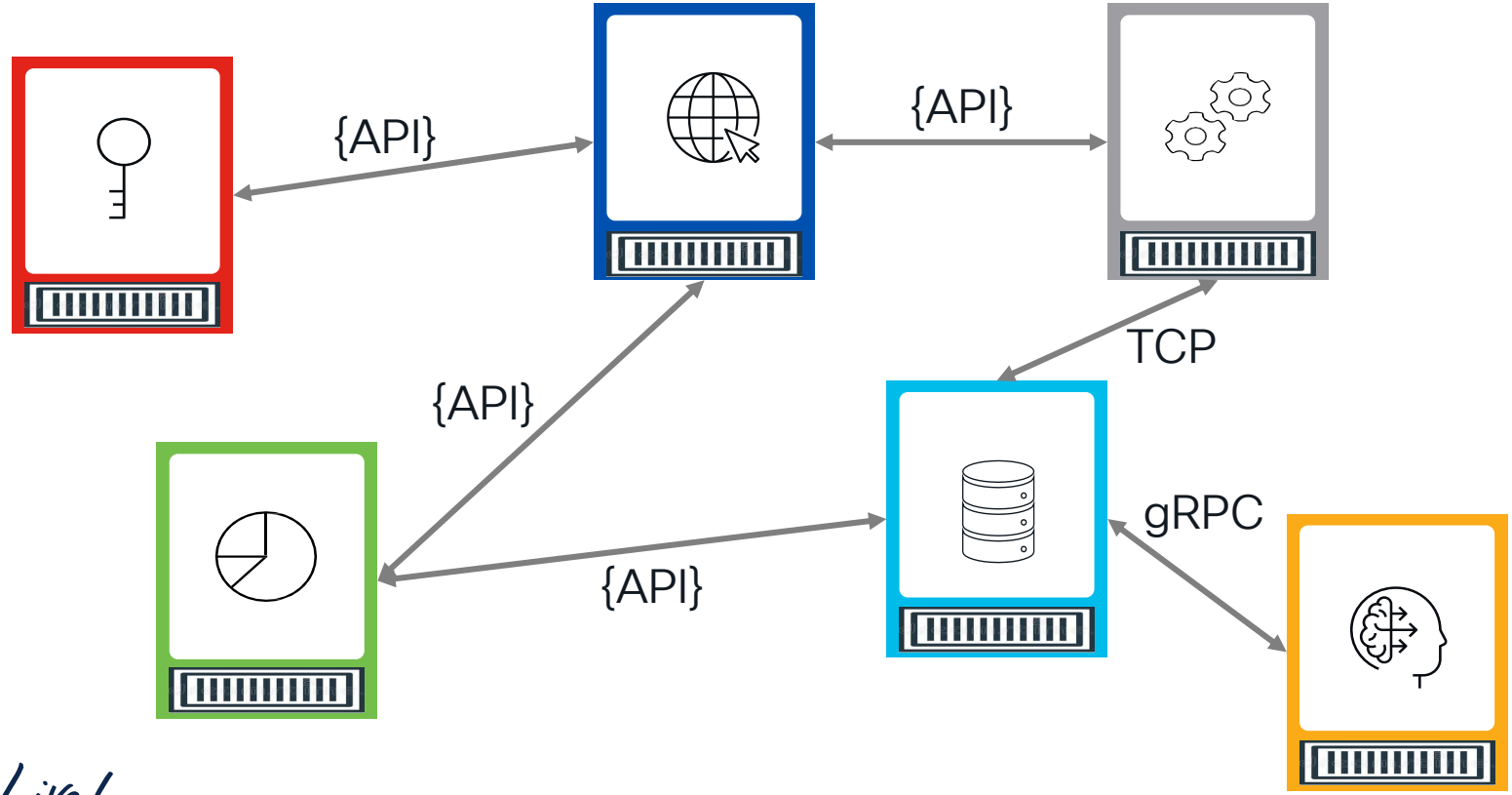
- Why do we need Service Mesh ?
- What is Service Mesh ?
- Multi-cluster Service Mesh Deployment Models
- Service Mesh Deployment Challenges
- Application Deployment in Multi-cluster Best Practices
- Demo
- Conclusion

Why do we need Service Mesh ?

Monolithic Application Architecture

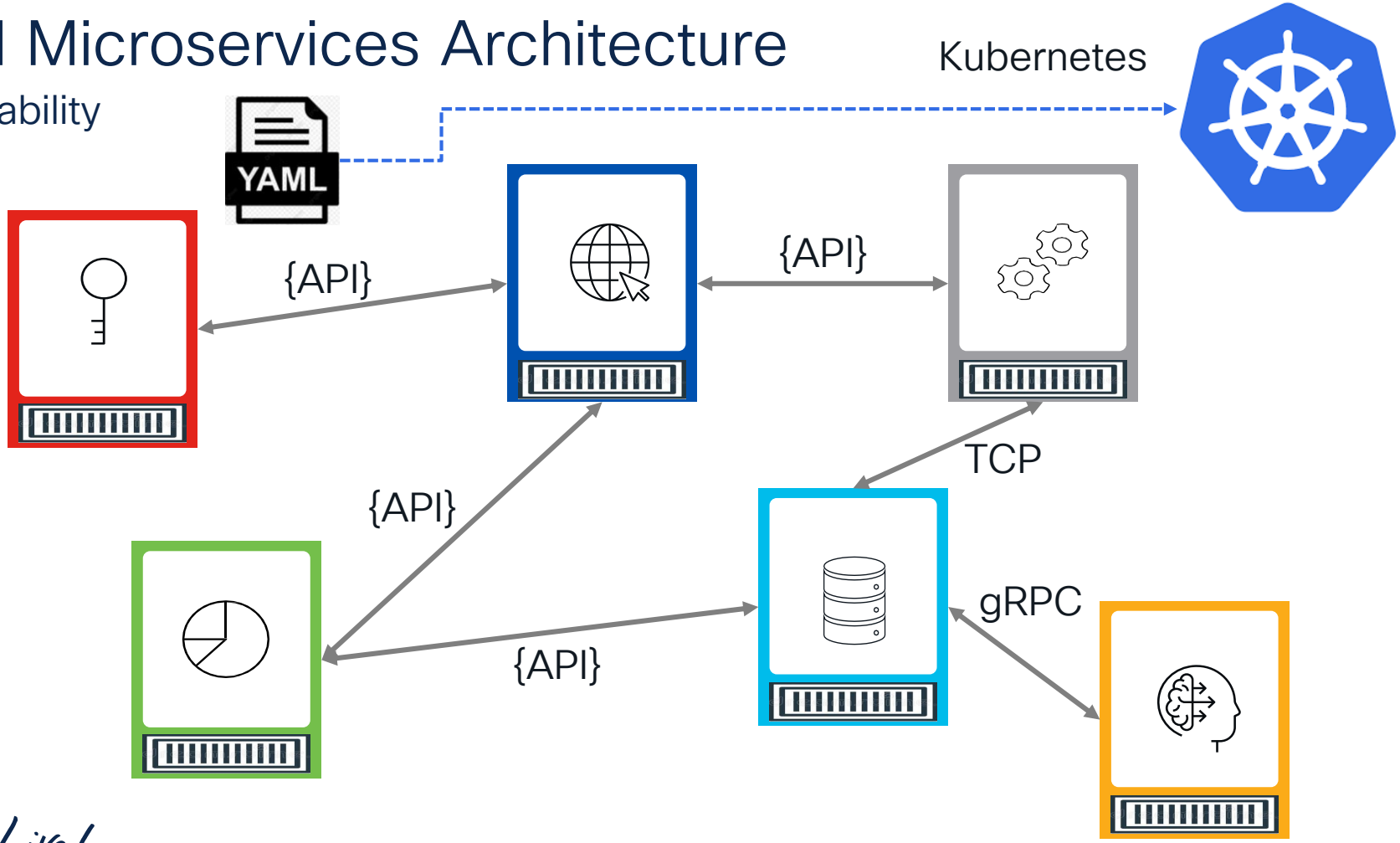


Cloud Native Microservices Architecture



CN Microservices Architecture

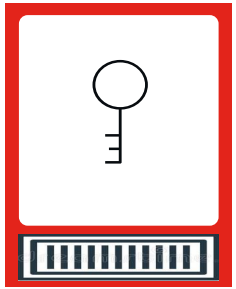
Scalability

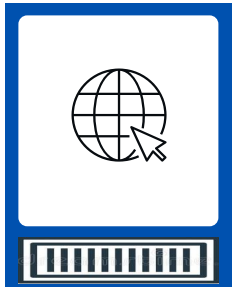


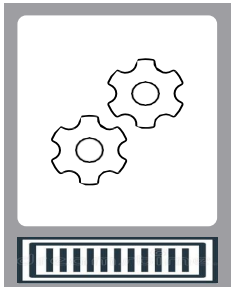
CN Microservices Architecture

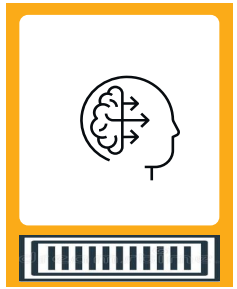
Scalability

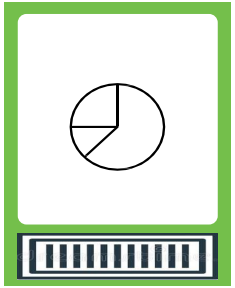
Kubernetes

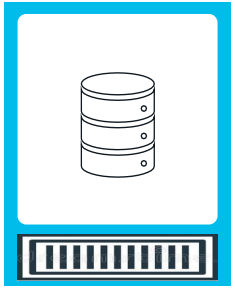








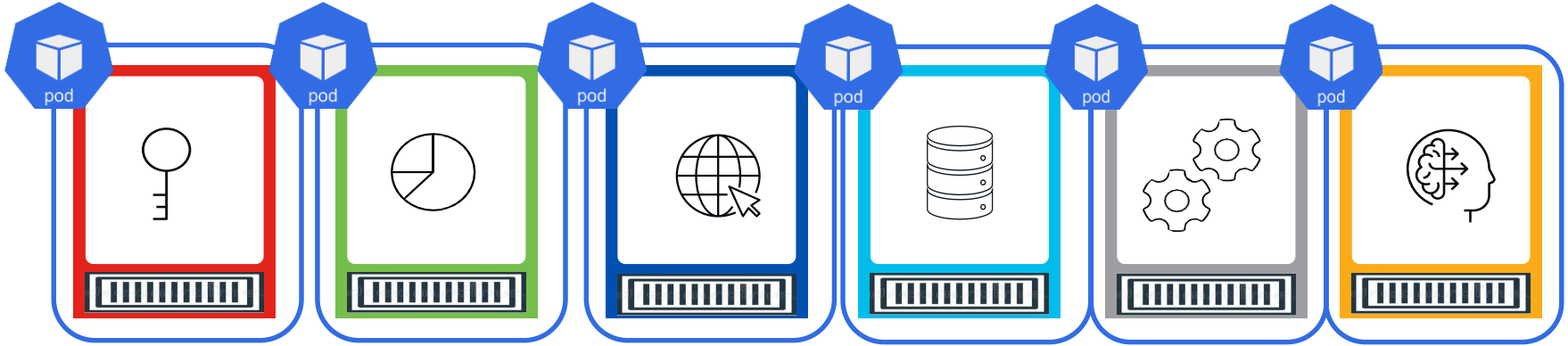




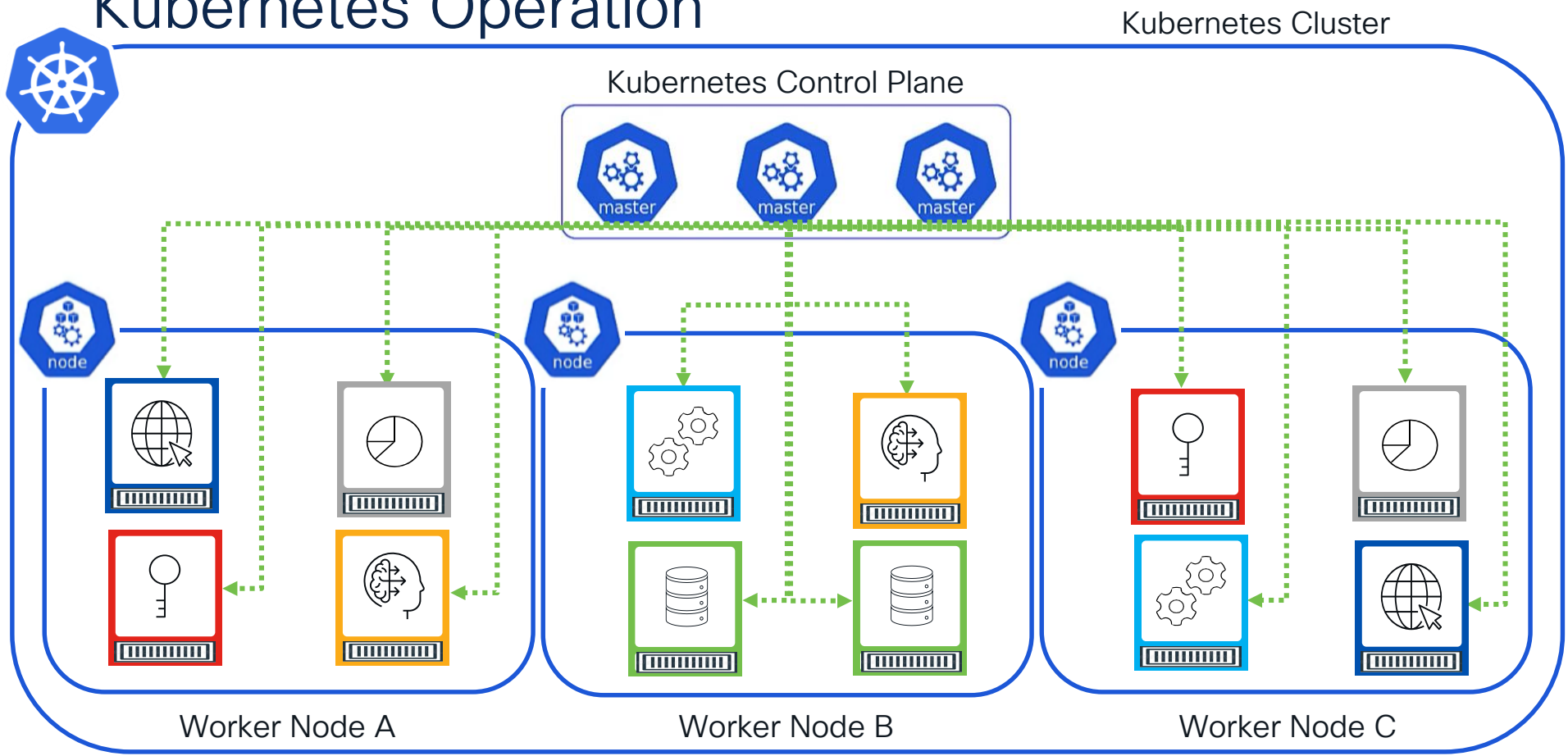
CN Microservices Architecture

Scalability

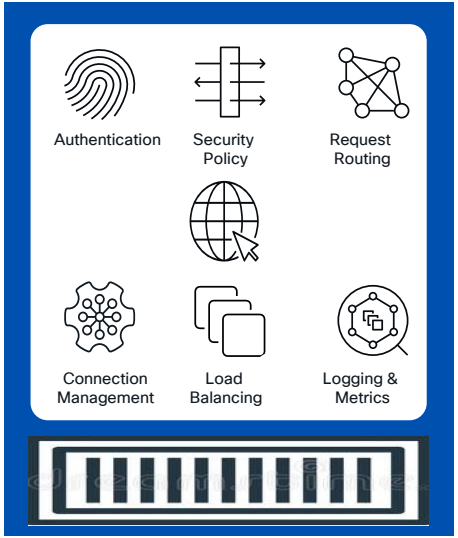
Kubernetes

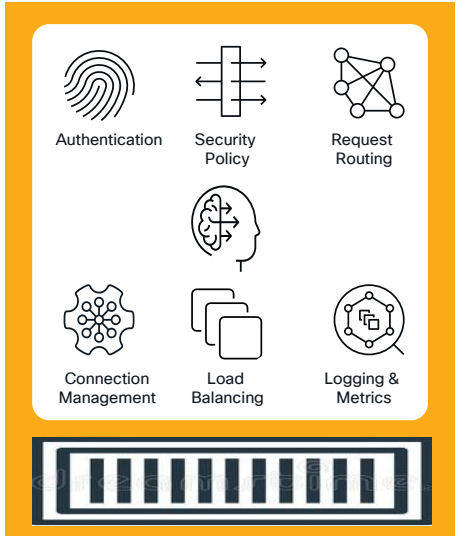


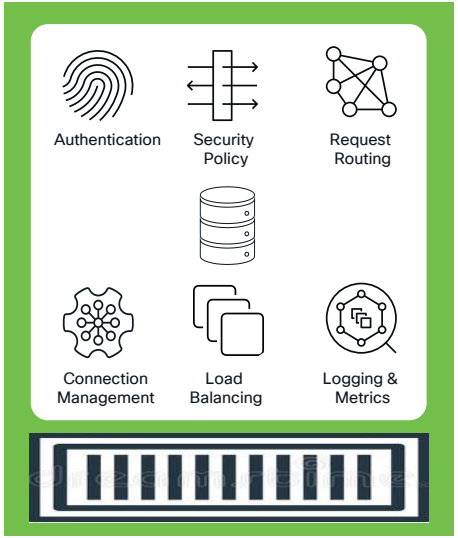
Kubernetes Operation



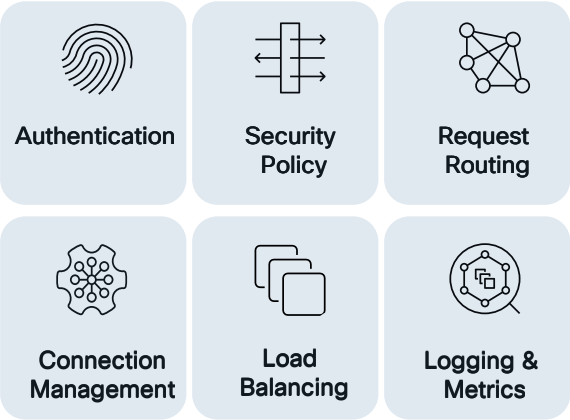
Microservice Common Functions

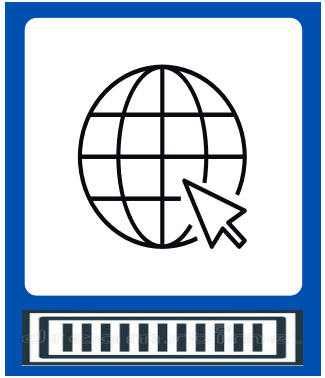


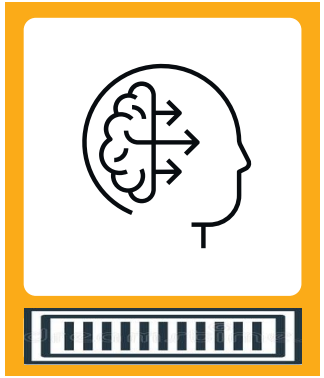


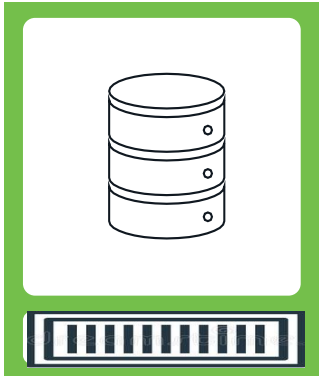


Microservice Common Functions



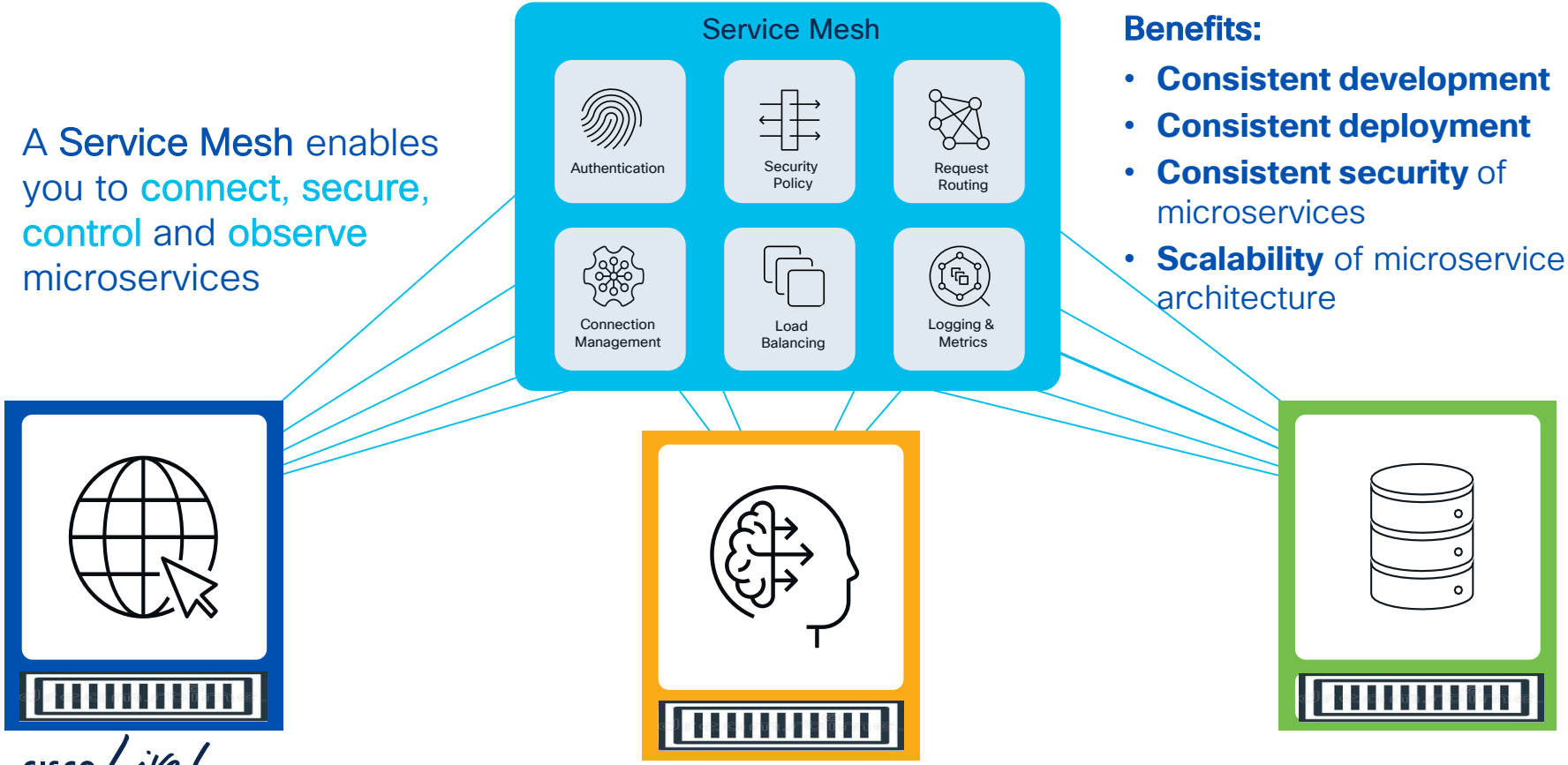






Service Mesh

A **Service Mesh** enables you to **connect**, **secure**, **control** and **observe** microservices



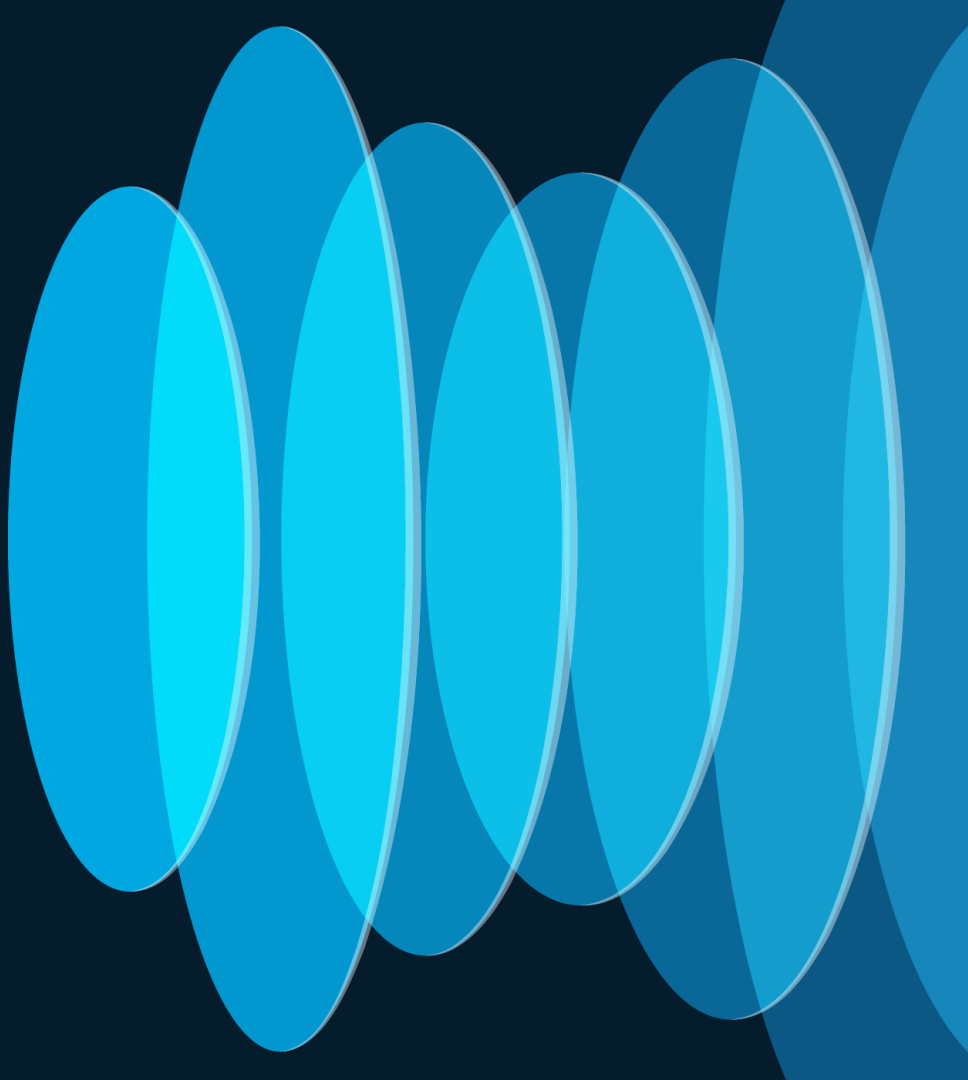
Benefits:

- **Consistent development**
- **Consistent deployment**
- **Consistent security** of microservices
- **Scalability** of microservice architecture

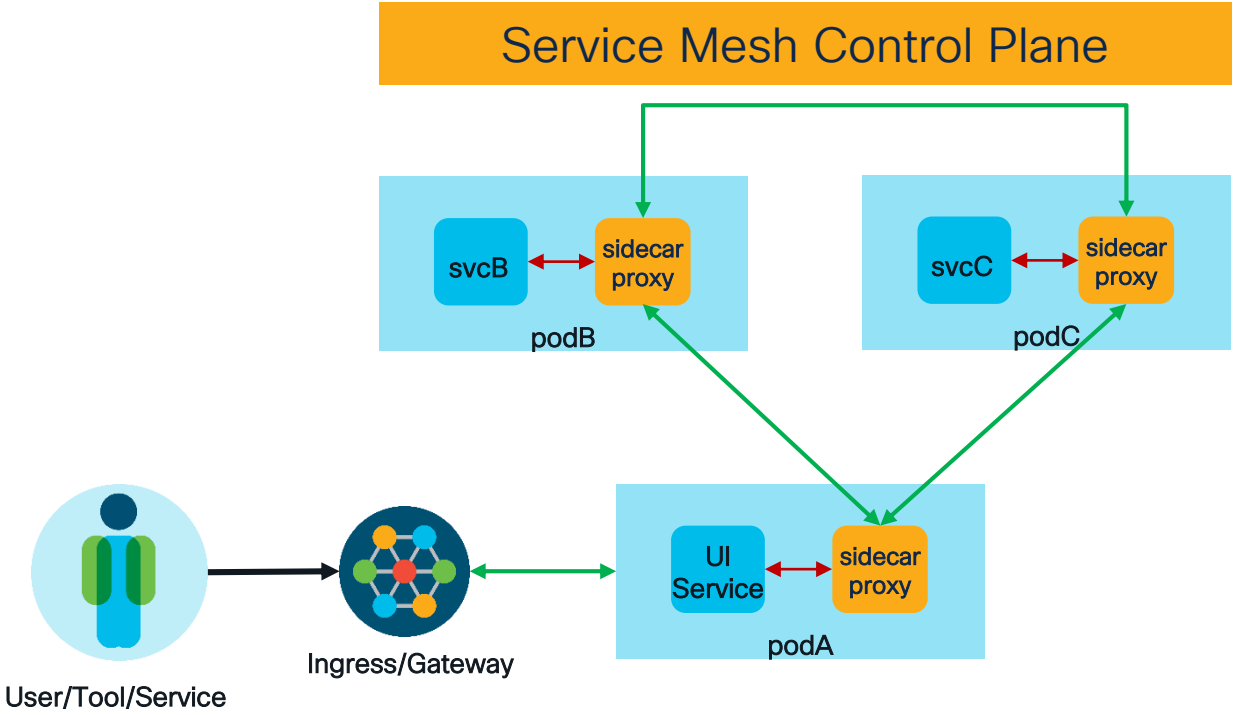
My Application Service Requirements

- I want to deploy a microservice
- I want to deploy using Kubernetes
- I have a bunch of requirements such as the need to handle:
  - Service failures
  - Retries
  - Circuit breaking
  - Topology changes
  - Monitoring
  - Tracing
  - Encryption between services
  - and more

What is a Service Mesh ?

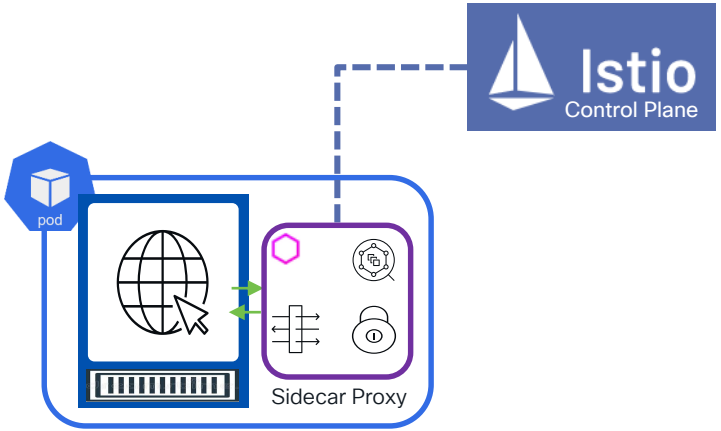


What is a Service Mesh?



- Infrastructure layer for service-to-service communication
- Can use a mesh of sidecar proxies:
  - Can inspect API transactions at Layer 7 and 4 (TCP)
  - Intelligent routing rules can be applied between endpoints

Sidecar Proxies and Service Mesh



- In a generic Kubernetes environment, a containerized application microservice is usually assigned to a dedicated pod
- However, several common service functions (such as observability, access policy, encryption, load-balancing, traffic management, etc.) can be standardized and enabled by creating a sidecar within the pod
- These common services are in turn centrally controlled by the service mesh control plane

Istio Overview

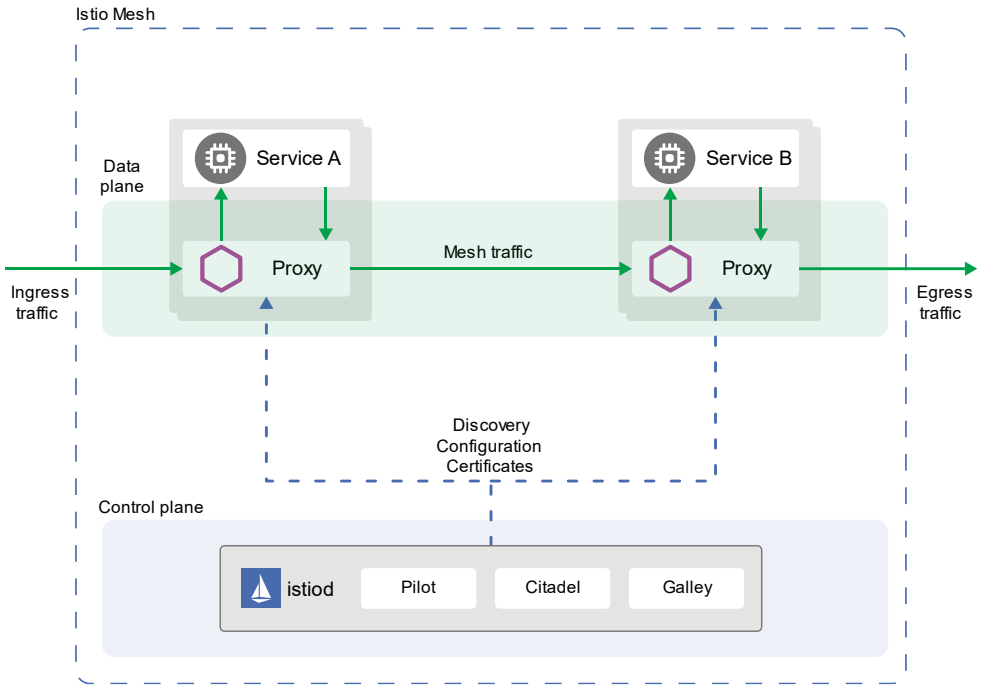
- An open-source project started by Google and IBM with help from the Envoy team at Lyft
  - https://istio.io/
  - https://github.com/istio
  - https://www.envoyproxy.io/
- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic
- Robust multicluster connectivity
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress
- Secure service-to-service authentication with strong identity assertions between services in a cluster

gRPC - Cross-platform Remote, Open Source, High Performace Remote Procedure Calls

Istio Architecture

https://istio.io/latest/docs/ops/deployment/architecture/

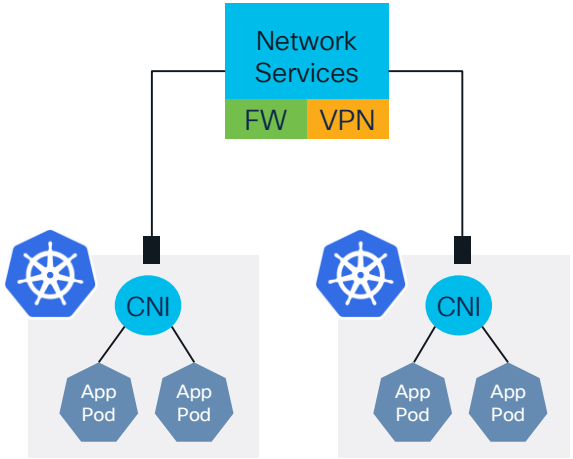
- istiod
  - Pilot
    - Handles service discovery and config data
    - Provides the Envoy proxies with the mesh topology and route rules
  - Galley
    - Validates user authored Istio API configuration on behalf of other control plane components
    - Top-level config ingestion, processing and distribution
  - Citadel
    - Provides certificates to the Envoy proxies for authentication and authorization
- Envoy
  - A proxy attached to every microservice
  - The connection point for a microservice to attach to the mesh



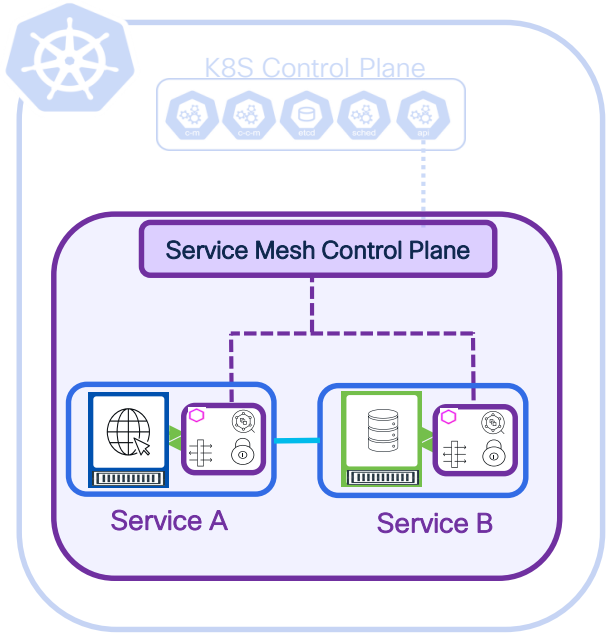
Multi-cluster Service Mesh Deployment Models

Istio Multicluster

- There are several reasons for establishing connectivity between Kubernetes clusters to include:
  - Service load balancing
  - Data replication
  - Service dependencies
  - Partner-provided service connectivity
  - etc..
- https://istio.io/latest/docs/ops/deployment/deployment-models/
  - Primary-Remote – single network
  - Primary-Remote – multiple networks
  - Multi-Primary – single network
  - **Multi-Primary – multiple networks**



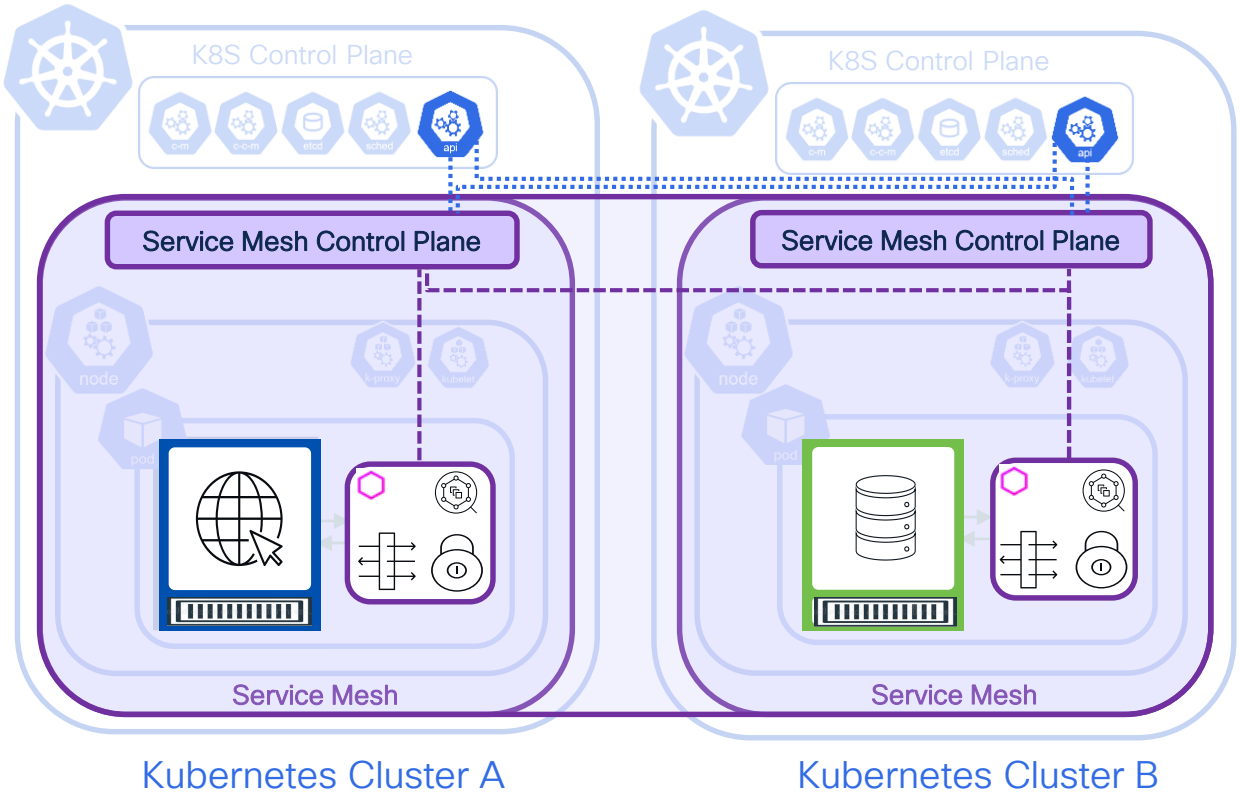
Single Cluster Deployment



Single Cluster

- Simplest Deployment
- Single Mesh/Control Plane
- Typically over same subnet
- End to end service visibility

Multi Cluster Deployment



- Multiple options
  - Single or Multiple Networks
  - Single or Multiple control planes
  - Zones or Regions
  - Distributed Applications
  - Loadbalancing and Istio Gateways

Single vs Multiple Networks

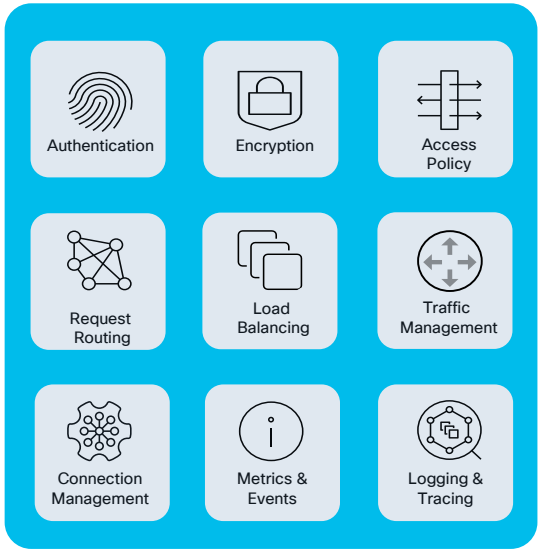
- Flat Networking
- Single subnet
- No overlapping IP
- Direct reachability between workloads without Istio gateways

- Overlapping IP or VIP ranges for **service endpoints**
- Crossing of administrative boundaries
- Fault tolerance
- Scaling of network addresses
- Compliance with standards that require network segmentation

Service Mesh Deployment Challenges

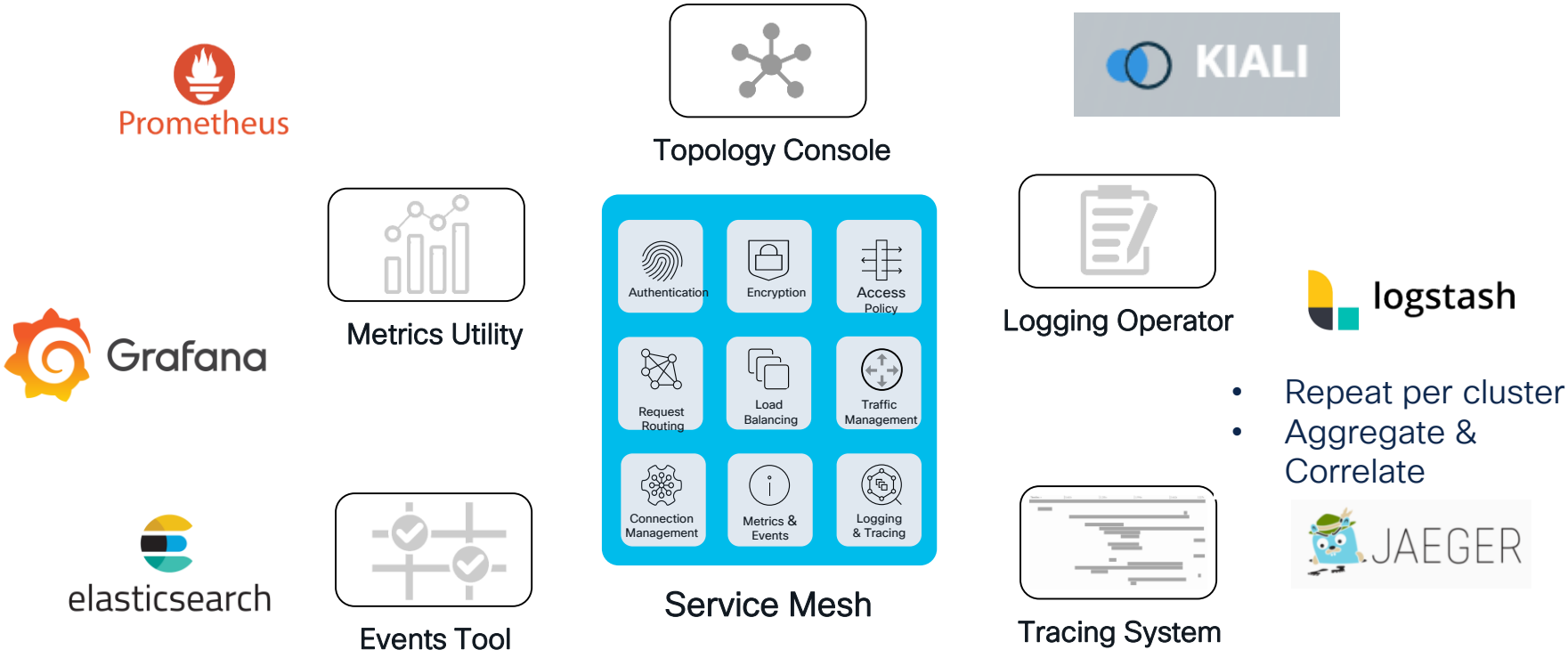
Service Mesh Deployment Challenges

- Lifecycle management
- Disparate/fragmented observability
- Multi-cluster challenges:
  - Availability
  - Cross-cluster service discovery
  - Inter-cluster traffic management policy
  - Multi-Tenancy



Service Mesh

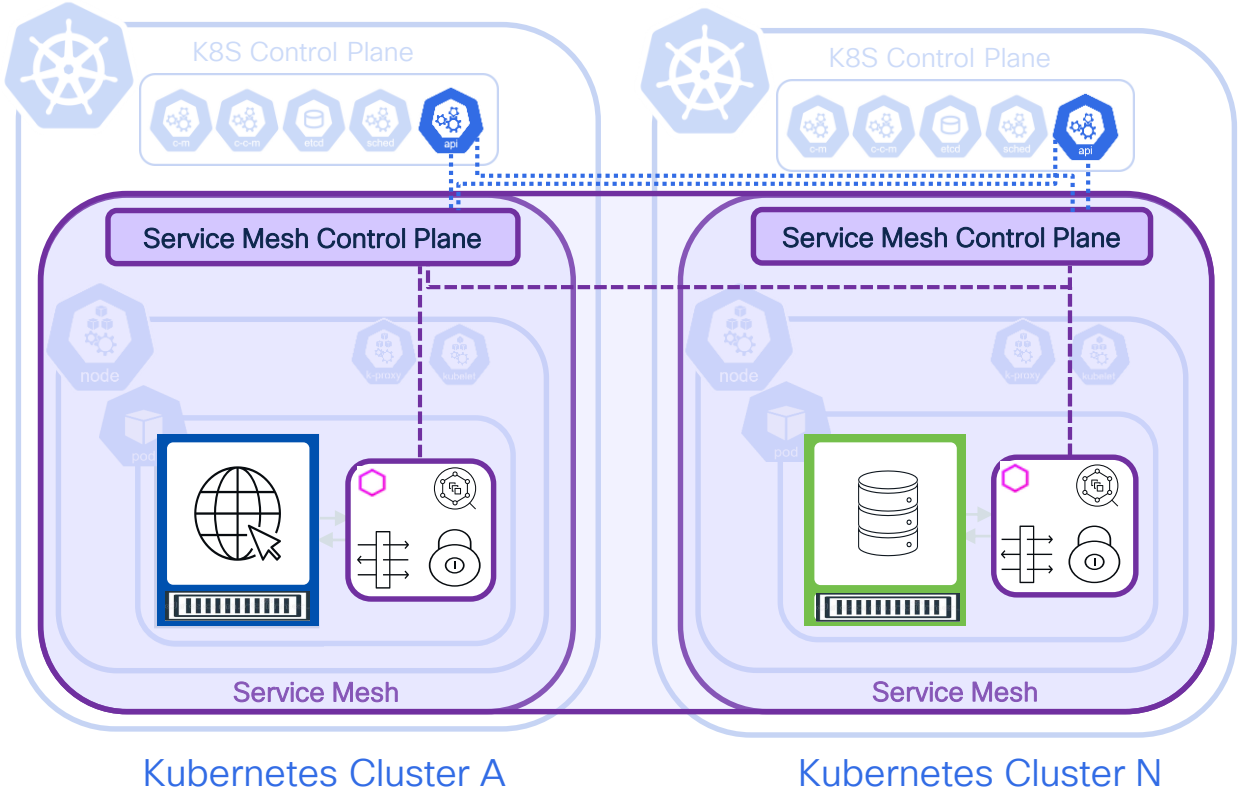
Service Mesh Observability Challenges



- Repeat per cluster
- Aggregate & Correlate

kubectl apply -f samples/addons

Enabling Multi-cluster Service Mesh



- Service meshes can be extended across clusters, such as by extending the control plane from a **primary** cluster to a **remote** cluster
  - Stable IP
  - Expose Control Plane via Istio GW
- Deploying multiple control planes across clusters, which is called a **multi-primary control plane**

Pre-planning

- Network CIDR
- Service Naming
- Establish Trust
- Enable DNS Proxy
- External Load balancer
- Expose services

Application Deployment in Multi-cluster Best Practices

Establish Trust – MUST for mTLS

- Have a dedicated ROOT CA
  - Make sure to download istio binary in both clusters curl -L https://istio.io/downloadIstio | sh -
  - Add istio folder to $PATH variable
- Setup Intermediate CA for each cluster
  - Copy cluster2 CA files to cluster 2

```
sudo scp cluster2/*.pem administrator@172.40.140.22:~/istio-1.21.2/certs/cluster2/
```

- Create secret in each cluster

Cluster 1

```
cd istio-1.21.2
mkdir -p certs
pushd certs
make -f ../tools/certs/Makefile.selfsigned.mk root-ca
make -f ../tools/certs/Makefile.selfsigned.mk cluster1-cacerts
!
make -f ../tools/certs/Makefile.selfsigned.mk cluster2-cacerts
```

Cluster 2

```
cd istio-1.21.2
mkdir -p certs
pushd certs
```

```
Cluster 1:

kubectl create secret generic cacerts -n istio-system      --from-file=cluster1
/ca-cert.pem       --from-file=cluster1/ca-key.pem       --from-file=cluster1/ro
ot-cert.pem       --from-file=cluster1/cert-chain.pem

Cluster 2:

kubectl create secret generic cacerts -n istio-system      --from-file=cluster2
/ca-cert.pem       --from-file=cluster2/ca-key.pem       --from-file=cluster2/ro
ot-cert.pem       --from-file=cluster2/cert-chain.pem
```

Enable DNS Proxy

- By default, Istio does not enable DNS proxy for services that are exposed to another cluster
- https://istio.io/latest/docs/ops/configuration/traffic-management/dns-proxy/#getting-started
- Without enabling DNS proxy, service in cluster 1 may not be resolvable on the 2nd cluster.

Add to the Istio Operator Config

```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    defaultConfig:
      proxyMetadata:
        # Enable basic DNS proxying
        ISTIO_META_DNS_CAPTURE: "true"
```

OR edit the config post-deployment

```
# kubectl edit istiocontrolplanes -n istio-system
  meshConfig:
    defaultConfig:
      . . .<output_summarized>
      proxyMetadata:
        ISTIO_META_ALS_ENABLED: "true"
        ISTIO_META_DNS_CAPTURE: "true"
        PROXY_CONFIG_XDS_AGENT: "true"
```

Application Load Balancers

- For Ingress, Egress and Eastwest gateways
- Typically available in Public Clouds
- For on-prem clusters, make sure to deploy load balancer for each cluster. For example MetalLB

kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.5/config/manifests/metallb-native.yaml

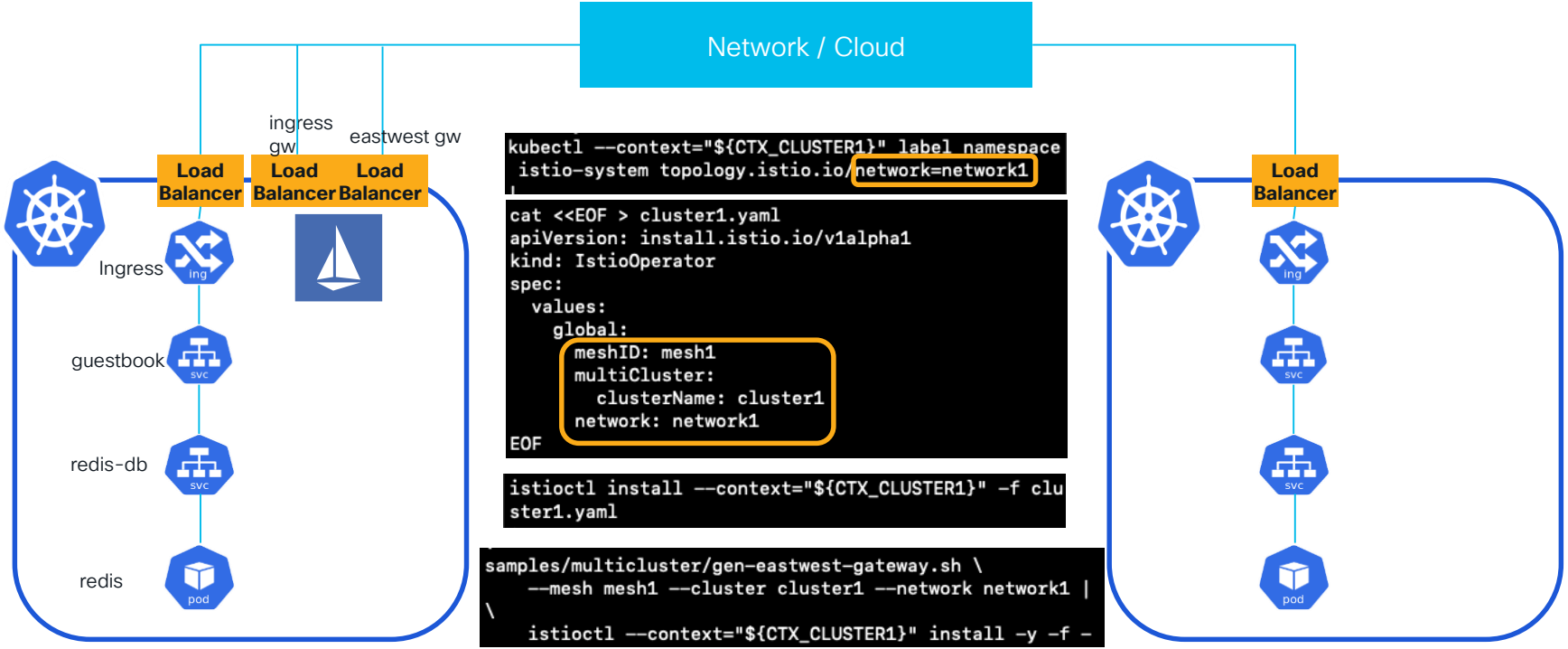
- Define IP Address Pools for each cluster
- Make sure these IPs are externally reachable for the GWs

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: lb-pool
  namespace: metallb-system
spec:
  addresses:
  - 172.40.143.181-172.40.143.190
```

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: example
  namespace: metallb-system
spec:
  ipAddressPools:
  - lb-pool
```

Istio Multi-Primary Deployment

Initialize Cluster1




```
kubectl --context="${CTX_CLUSTER1}" label namespace
 istio-system topology.istio.io/network=network1
```

```
cat <<EOF > cluster1.yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    global:
      meshID: mesh1
      multiCluster:
        clusterName: cluster1
      network: network1
EOF
```

```
istioctl install --context="${CTX_CLUSTER1}" -f clu
ster1.yaml
```

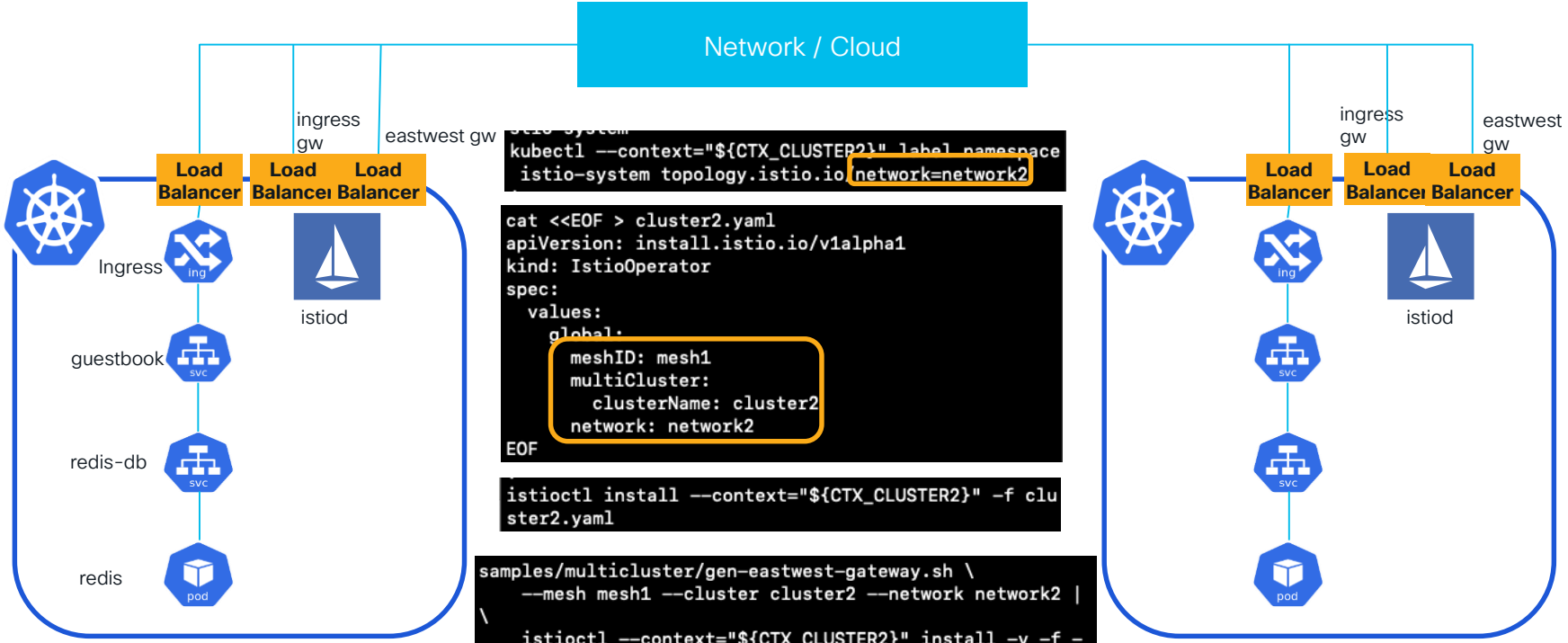
```
samples/multicluster/gen-eastwest-gateway.sh \
    --mesh mesh1 --cluster cluster1 --network network1 |
\
    istioctl --context="${CTX_CLUSTER1}" install -y -f -
```

cluster1
context: cluster1-admin@cluster1

cluster2
context: cluster2-admin@cluster2

Istio Multi-Primary Deployment

Initialize Cluster2




```
kubectl --context="${CTX_CLUSTER2}" label namespace
 istio-system topology.istio.io/network=network2
```

```
cat <<EOF > cluster2.yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    global:
      meshID: mesh1
      multiCluster:
        clusterName: cluster2
      network: network2
EOF
```

```
istioctl install --context="${CTX_CLUSTER2}" -f clu
ster2.yaml
```

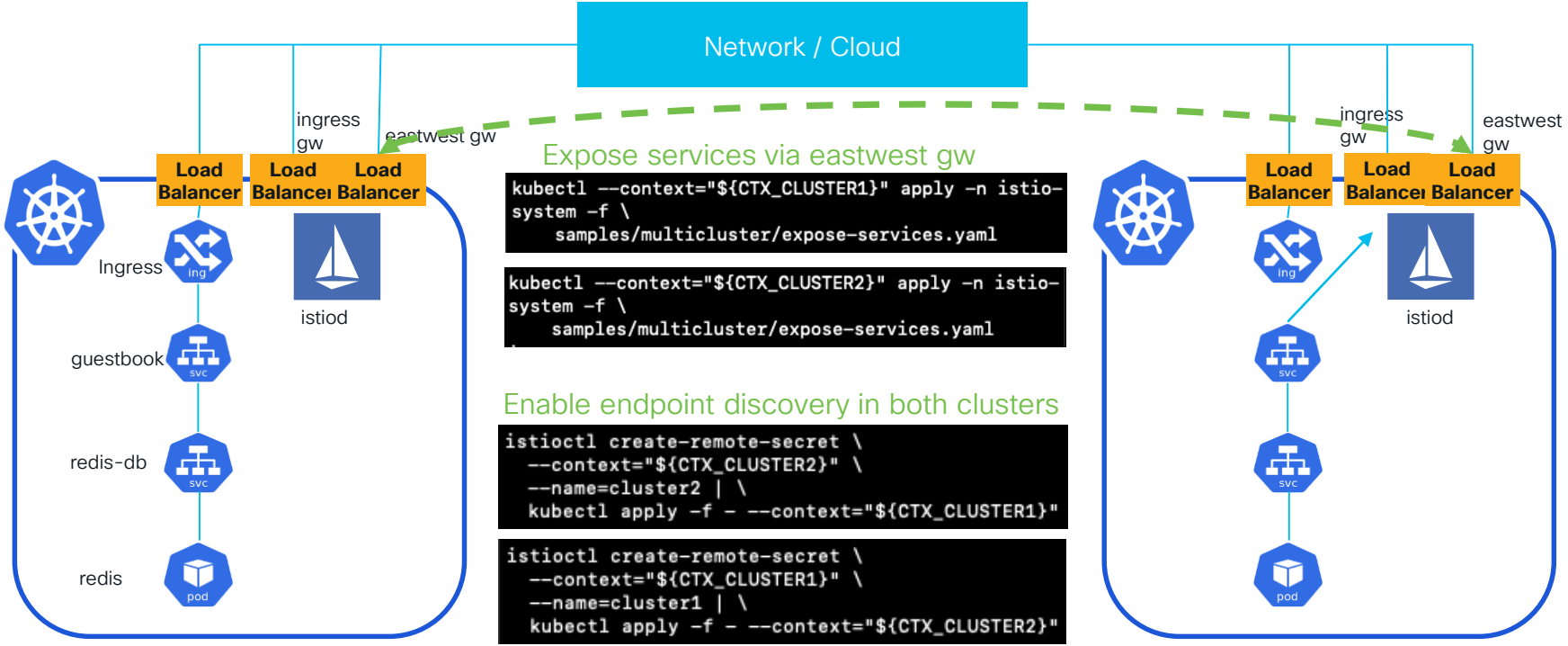
```
samples/multicluster/gen-eastwest-gateway.sh \
    --mesh mesh1 --cluster cluster2 --network network2 |
\
    istioctl --context="${CTX_CLUSTER2}" install -y -f -
```

cluster1
context: cluster1-admin@cluster1

cluster2
context: cluster2-admin@cluster2

Istio Multi-Primary Deployment

Expose services and Enable endpoint discovery



Expose services via eastwest gw

```
kubectl --context="${CTX_CLUSTER1}" apply -n istio-system -f \
    samples/multicluster/expose-services.yaml
```

```
kubectl --context="${CTX_CLUSTER2}" apply -n istio-system -f \
    samples/multicluster/expose-services.yaml
```

Enable endpoint discovery in both clusters

```
istioctl create-remote-secret \
  --context="${CTX_CLUSTER2}" \
  --name=cluster2 | \
  kubectl apply -f - --context="${CTX_CLUSTER1}"
```

```
istioctl create-remote-secret \
  --context="${CTX_CLUSTER1}" \
  --name=cluster1 | \
  kubectl apply -f - --context="${CTX_CLUSTER2}"
```

cluster1
context: cluster1-admin@cluster1

cluster2
context: cluster2-admin@cluster2

Mesh status – Cluster 1

```
administrator@cl1-istio-master:~/istio-1.21.2$ istioctl remote-clusters -c ~/.kube/config
NAME          SECRET                                          STATUS     ISTIOD
cluster2      istio-system/istio-remote-secret-cluster2       synced     istiod-6696b6844d-cwj44
```

```
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl get pods -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
istio-eastwestgateway-c89658c74-6sq8b   1/1     Running   0          103m
istio-ingressgateway-7b4fdf6d69-lj2ds   1/1     Running   0          104m
istiod-6696b6844d-cwj44                 1/1     Running   0          104m
administrator@cl1-istio-master:~/istio-1.21.2$
administrator@cl1-istio-master:~/istio-1.21.2$
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl get svc -n istio-system
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)                                                    AGE
istio-eastwestgateway  LoadBalancer   10.100.159.133   172.40.143.182   15021:31046/TCP,15443:32163/TCP,15012:30919/TCP,15017:32420/TCP   103m
istio-ingressgateway   LoadBalancer   10.102.23.142    172.40.143.181   15021:31555/TCP,80:30553/TCP,443:32092/TCP                 105m
istiod                 ClusterIP      10.105.208.143   <none>           15010/TCP,15012/TCP,443/TCP,15014/TCP                      105m
administrator@cl1-istio-master:~/istio-1.21.2$
```

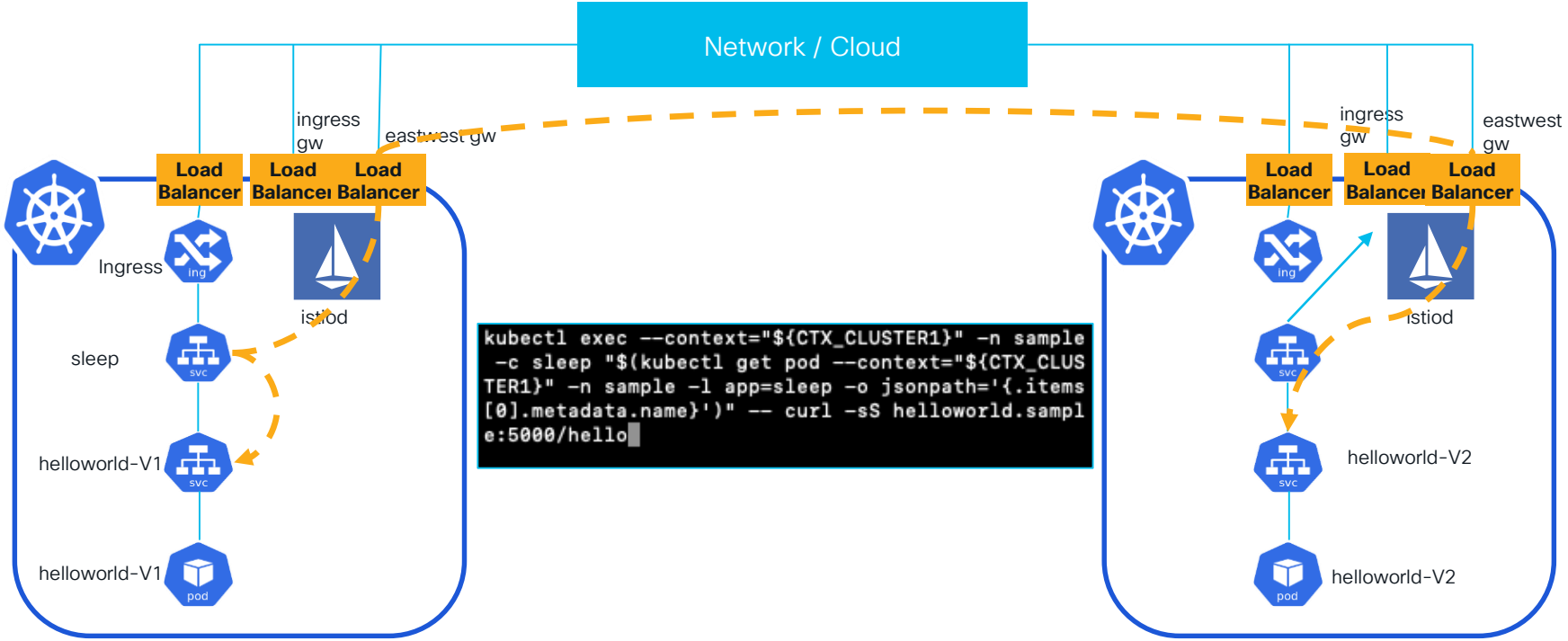
Mesh status – Cluster 2

```
administrator@multi-primary-master:~/istio-1.21.2$ istioctl remote-clusters
NAME          SECRET                                        STATUS     ISTIOD
cluster1      istio-system/istio-remote-secret-cluster1     synced     istiod-6c4d88c6d7-wpzz8
administrator@multi-primary-master:~/istio-1.21.2$
```

```
administrator@multi-primary-master:~/istio-1.21.2$ kubectl get pods -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
istio-eastwestgateway-7cbc7b94b8-pmkc7  1/1     Running   0          56m
istio-ingressgateway-6dbdffc56f-v8vz8   1/1     Running   0          58m
istiod-6c4d88c6d7-wpzz8                 1/1     Running   0          58m
administrator@multi-primary-master:~/istio-1.21.2$
administrator@multi-primary-master:~/istio-1.21.2$
administrator@multi-primary-master:~/istio-1.21.2$ kubectl get svc -n istio-system
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)                                                    AGE
istio-eastwestgateway   LoadBalancer   10.96.31.100     172.40.143.192   15021:31590/TCP,15443:31841/TCP,15012:31272/TCP,15017:32063/TCP   56m
istio-ingressgateway    LoadBalancer   10.108.147.230   172.40.143.191   15021:32147/TCP,80:31216/TCP,443:32293/TCP                 58m
istiod                  ClusterIP      10.108.239.229   <none>           15010/TCP,15012/TCP,443/TCP,15014/TCP                      58m
administrator@multi-primary-master:~/istio-1.21.2$
```

Deploy Application

Validate cross cluster connectivity




```
kubectl exec --context="${CTX_CLUSTER1}" -n sample
 -c sleep "$(kubectl get pod --context="${CTX_CLUS
TER1}" -n sample -l app=sleep -o jsonpath='{.items
[0].metadata.name}')" -- curl -sS helloworld.sampl
e:5000/hello
```

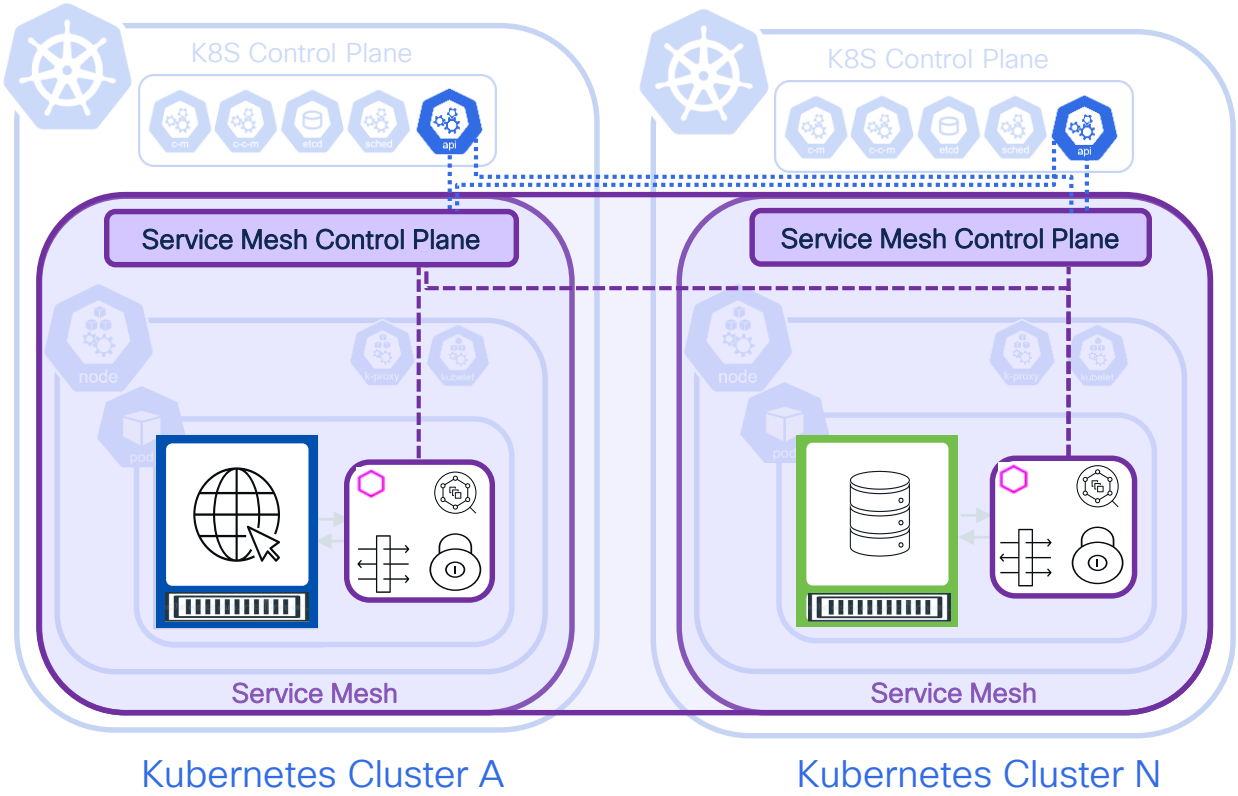
cluster1
context: cluster1-admin@cluster1

cluster2
context: cluster2-admin@cluster2

Deploy application in Multi-cluster

```
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl get pods -n sample -o wide
NAME                            READY   STATUS    RESTARTS   AGE     IP                NODE              NOMINATED NODE   READINESS GATES
helloworld-v1-867747c89-wzjch   2/2     Running   0          9m43s   192.168.184.139   cl1-istio-work2   <none>           <none>
sleep-7656cf8794-qmx9p          2/2     Running   0          8m2s    192.168.138.131   cl1-istio-work3   <none>           <none>
administrator@cl1-istio-master:~/istio-1.21.2$
administrator@cl1-istio-master:~/istio-1.21.2$
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl get svc -n sample
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
helloworld   ClusterIP   10.105.200.149   <none>        5000/TCP   11m
sleep        ClusterIP   10.96.147.41     <none>        80/TCP     8m15s
administrator@cl1-istio-master:~/istio-1.21.2$
administrator@cl1-istio-master:~/istio-1.21.2$
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl exec -n sample -c sleep "$(kubectl get pod --context="${CTX_CLUSTER1}" -n sample -l app=sleep -o jsonpath='{
.items[0].metadata.name}')" -- curl -sS helloworld.sample:5000/hello
Hello version: v1, instance: helloworld-v1-867747c89-wzjch
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl exec -n sample -c sleep "$(kubectl get pod --context="${CTX_CLUSTER1}" -n sample -l app=sleep -o jsonpath='{
.items[0].metadata.name}')" -- curl -sS helloworld.sample:5000/hello
Hello version: v1, instance: helloworld-v1-867747c89-wzjch
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl exec -n sample -c sleep "$(kubectl get pod --context="${CTX_CLUSTER1}" -n sample -l app=sleep -o jsonpath='{
.items[0].metadata.name}')" -- curl -sS helloworld.sample:5000/hello
Hello version: v2, instance: helloworld-v2-7f46498c69-fg5p7
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl exec -n sample -c sleep "$(kubectl get pod --context="${CTX_CLUSTER1}" -n sample -l app=sleep -o jsonpath='{
.items[0].metadata.name}')" -- curl -sS helloworld.sample:5000/hello
Hello version: v1, instance: helloworld-v1-867747c89-wzjch
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl exec -n sample -c sleep "$(kubectl get pod --context="${CTX_CLUSTER1}" -n sample -l app=sleep -o jsonpath='{
.items[0].metadata.name}')" -- curl -sS helloworld.sample:5000/hello
Hello version: v1, instance: helloworld-v1-867747c89-wzjch
administrator@cl1-istio-master:~/istio-1.21.2$ kubectl exec -n sample -c sleep "$(kubectl get pod --context="${CTX_CLUSTER1}" -n sample -l app=sleep -o jsonpath='{
.items[0].metadata.name}')" -- curl -sS helloworld.sample:5000/hello
Hello version: v2, instance: helloworld-v2-7f46498c69-fg5p7
administrator@cl1-istio-master:~/istio-1.21.2$
```

Enabling a Multi-Primary Control Plane

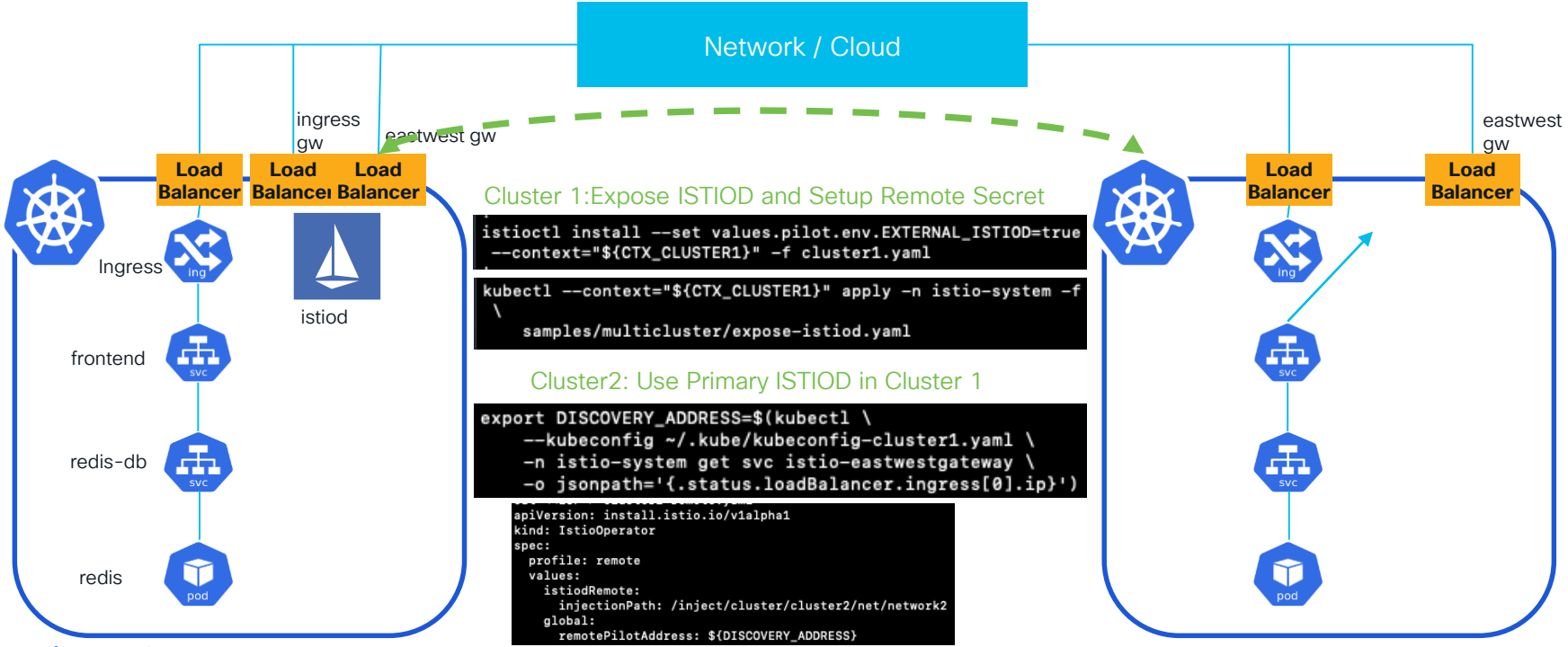


Benefits :

- Limited Scope
  - Cluster specific Configuration changes
  - Cluster specific impact if control plane is unavailable
  - Controlled Configuration rollout
- Service isolation/limited visibility
- High availability
- Cross-cluster endpoint/service discovery

Istio Primary-Remote Deployment

Key Differences



Cluster 1:Expose ISTIOD and Setup Remote Secret

```
istioctl install --set values.pilot.env.EXTERNAL_ISTIOD=true
 --context="${CTX_CLUSTER1}" -f cluster1.yaml
```

```
kubectl --context="${CTX_CLUSTER1}" apply -n istio-system -f
 \
    samples/multicluster/expose-istiod.yaml
```

Cluster2: Use Primary ISTIOD in Cluster 1

```
export DISCOVERY_ADDRESS=$(kubectl \
    --kubeconfig ~/.kube/kubeconfig-cluster1.yaml \
    -n istio-system get svc istio-eastwestgateway \
    -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
```

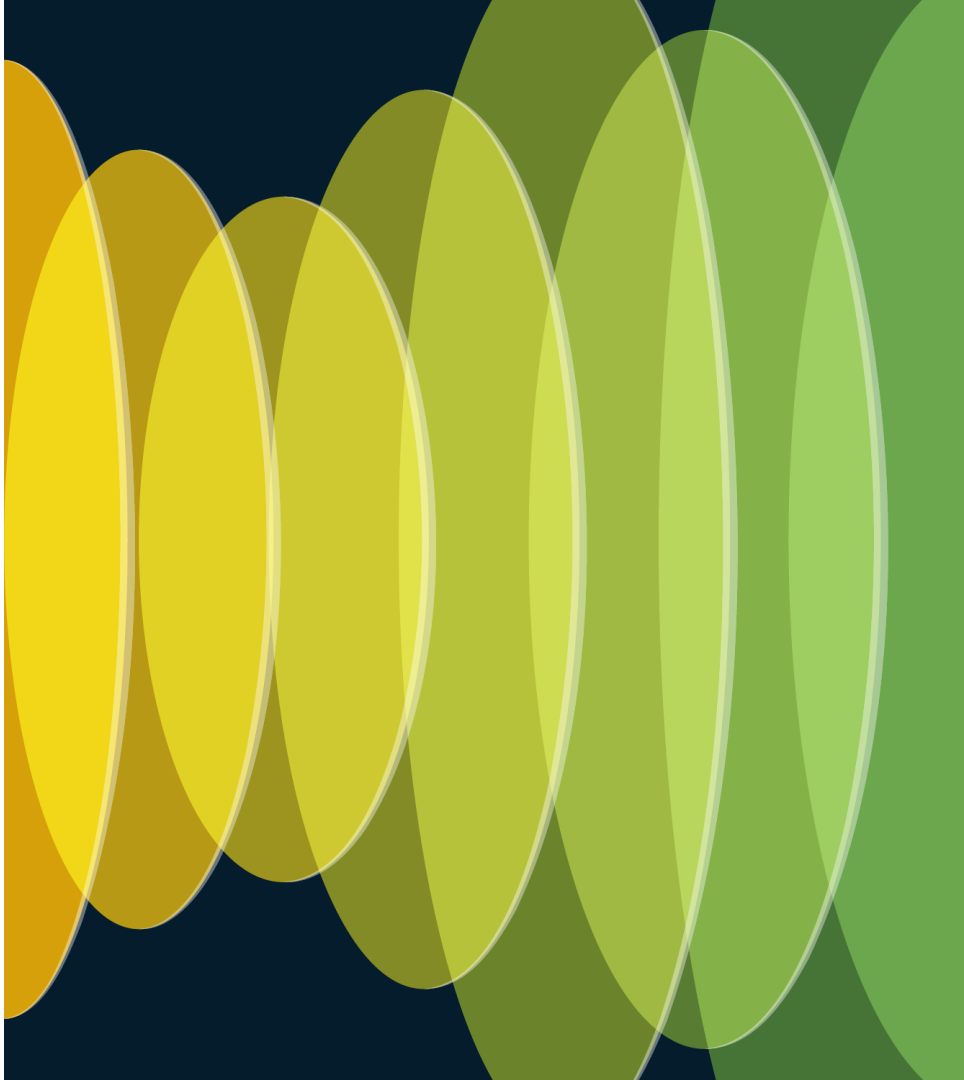
```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  profile: remote
  values:
    istiodRemote:
      injectionPath: /inject/cluster/cluster2/net/network2
    global:
      remotePilotAddress: ${DISCOVERY_ADDRESS}
```

```
istioctl install --context="${CTX_CLUSTER2}" -f cluster2-rem
ote.yaml
```

cluster1
context: cluster1-admin@cluster1

cluster2
context: cluster2-admin@cluster2

Demo



Conclusion

- Deployment options exist for Multi cluster Service Mesh
- Single vs Multi Network Deployment
- Certificate Setup
- DNS Proxy Setup
- Load balancers and Gateways
- Cross-cluster Service Discovery

Complete Your Session Evaluations

Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to **win 1 of 5 full conference passes** to Cisco Live 2025.

Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

Level up and earn **exclusive prizes!**

Complete your surveys in the **Cisco Live mobile app.**

Continue your education

- All about Istio Service Mesh: https://istio.io/latest/docs/
- Istio Examples: https://istio.io/latest/docs/examples/microservices-istio/
- Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs
- Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me at: Webex App - https://ciscolive.ciscoevents.com/ciscolivebot/#BRKCLD-2019

Thank you

Verification of Endpoints in Cluster 1

```
administrator@multi-primary-master:~/istio-1.21.2$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
loki-0                              0/2     Pending   0          10d
ratings-v1-6484c4d9bb-25cnf         2/2     Running   0          79m
reviews-v3-5b9bd44f4-ztxsd          2/2     Running   0          79m
```

```
administrator@multi-primary-master:~/istio-1.21.2$ istioctl proxy-config endpoints ratings-v1-6484c4d9bb-25cnf --cluster "outbound|9080||productpage.default.svc.cluster.local"
ENDPOINT                STATUS      OUTLIER CHECK     CLUSTER
172.40.143.182:15443    HEALTHY     OK                outbound|9080||productpage.default.svc.cluster.local
administrator@multi-primary-master:~/istio-1.21.2$
administrator@multi-primary-master:~/istio-1.21.2$ istioctl proxy-config endpoints ratings-v1-6484c4d9bb-25cnf --cluster "outbound|9080||details.default.svc.cluster.local"
ENDPOINT                STATUS      OUTLIER CHECK     CLUSTER
172.40.143.182:15443    HEALTHY     OK                outbound|9080||details.default.svc.cluster.local
administrator@multi-primary-master:~/istio-1.21.2$
administrator@multi-primary-master:~/istio-1.21.2$ istioctl proxy-config endpoints ratings-v1-6484c4d9bb-25cnf --cluster "outbound|9080||reviews.default.svc.cluster.local"
ENDPOINT                STATUS      OUTLIER CHECK     CLUSTER
172.40.143.182:15443    HEALTHY     OK                outbound|9080||reviews.default.svc.cluster.local
192.168.179.215:9080    HEALTHY     OK                outbound|9080||reviews.default.svc.cluster.local
```