cisco Live!

ALL IN

#CiscoLive

The bridge to possible

Understanding Multicluster Kubernetes Connectivity Options

Shannon McFarland – CCIE #5245
Distinguished Engineer
@eyepv6

BRKETI-2003

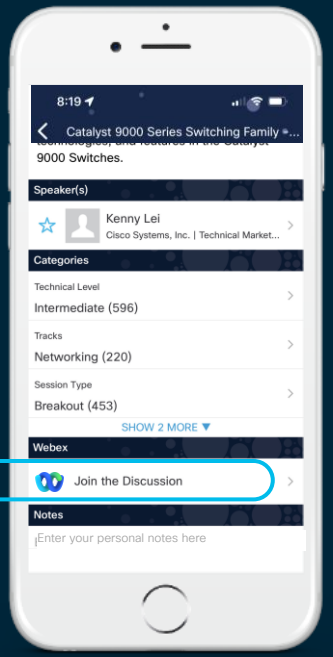
Cisco Webex App

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



https://ciscolive.ciscoevents.com/ciscolivebot/#BRKETI-2003

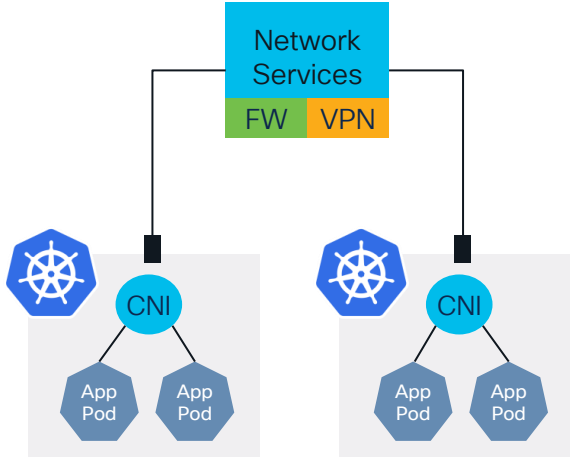
Webex spaces will be moderated by the speaker until June 17, 2022.

Agenda

- Multicluster Connectivity – Pain points
- Kubernetes Services, Ingress, Load Balancer
- Cilium Cluster Mesh
- Service Mesh
  - Linkerd
  - Istio

Multicluster Connectivity – What is all the fuss about?

- There are several reasons for establishing connectivity between Kubernetes clusters to include:
  - Service load balancing
  - Data replication
  - Service dependencies
  - Partner-provided service connectivity
  - etc..
- Today, many assumptions are made about the underlying infrastructure that exists underneath and in between these clusters:
  - Use ingress/load balancers and let basic networking and name resolution sort it out
  - Intra-VPC/Intra-network – Deploy the clusters in the same VPCs/networks to facilitate easier connectivity
  - Inter-VPC/Inter-network – Networking is already built and managed (Hybrid cloud, VPC peering, etc.)
- Regardless of the assumptions or justification, something and someone has to deal with service-to-service connectivity – let's explore some options

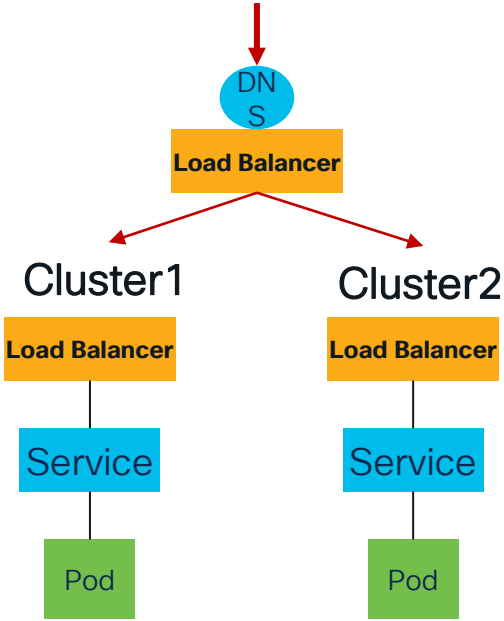


Multicluster Connectivity – Options Galore!

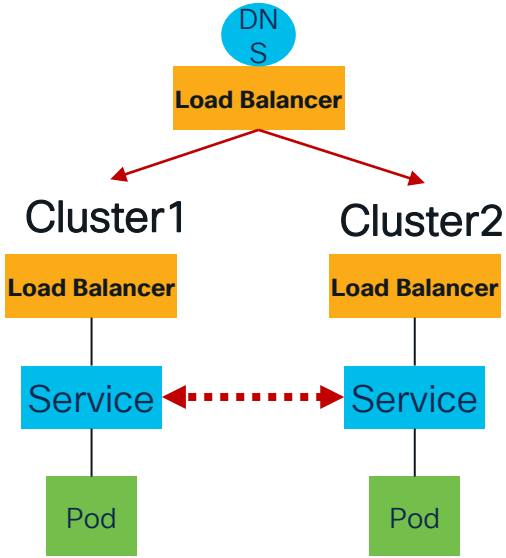
- There are many solutions for linking workloads that are hosted on different Kubernetes clusters – let's look at a few of them
- CNI-based
  - Cilium Cluster Mesh – Global load balancing is great – Service-to-service can be dicey
- Gateway-based
  - Submariner – A Layer 3/4 centric approach – Service-to-service is a strength – Not the smoothest implementation
- Application Service Mesh-based – Layer 4/7 networking, robust security and observability
  - Linkerd
  - Istio

A Couple of Use Cases

Global Load Balancing



Service-to-Service/Pod-to-Pod

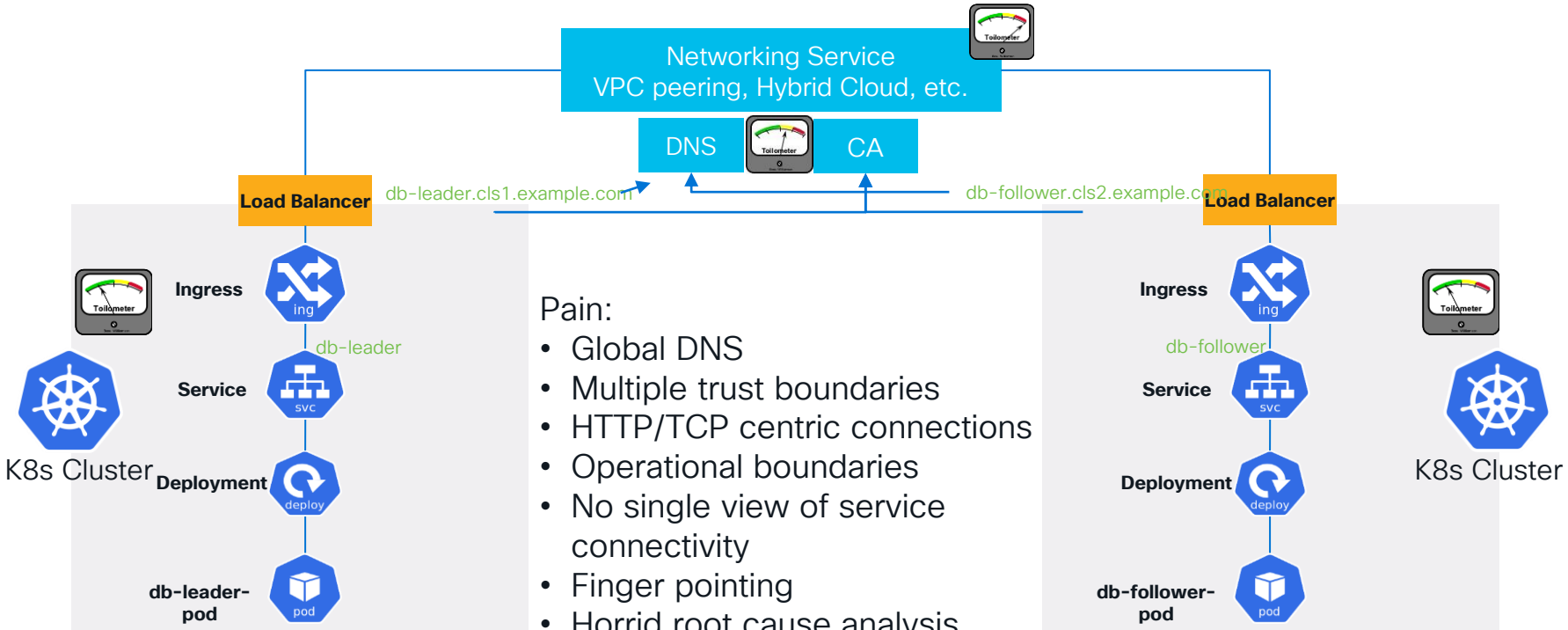


A Note About Shared Service Naming

	Cilium Cluster Mesh	Submariner	Linkerd	Istio	Cisco Calisti (Istio)
Service Name	Unchanged – Global LB Custom service – Create phantom/ghost service for service-to-service use case	<SVC>.default.svc.clusterset.local	<SVC-CLUSTER>-.default.svc.cluster.local	Unchanged – Global LB Custom service – Create phantom/ghost service for service-to-service use case	Same as Istio
Example	my-custom-service-name.default.svc.cluster.local	redis-cart.default.svc.clusterset.local	redis-cart-cluster1.default.svc.cluster.local	my-custom-service-name.default.svc.cluster.local	Same as Istio
Special Config	kind: Service metadata: name: redis-cart annotations: io.cilium/global-service: "true"				

Kubernetes Services, Ingresses, Load Balancers

K8s Multicluster Connectivity – Using Ingress, Services, LBs



Pain:

- Global DNS
- Multiple trust boundaries
- HTTP/TCP centric connections
- Operational boundaries
- No single view of service connectivity
- Finger pointing
- Horrid root cause analysis

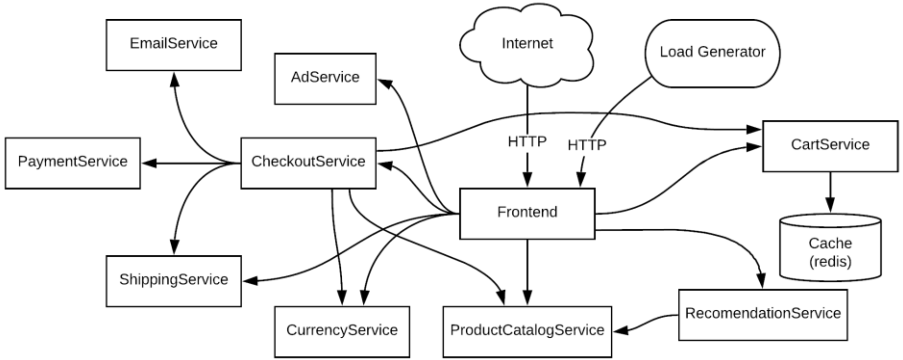
Cilium CNI + Cilium Cluster Mesh

Cilium Cluster Mesh

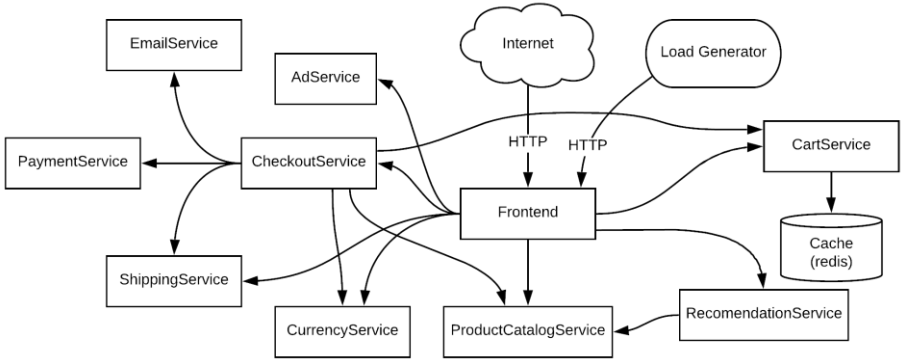
- eBPF-based Networking, Observability and Security:
  - https://cilium.io/
  - A CNCF project
- Cilium Cluster Mesh: https://docs.cilium.io/en/stable/gettingstarted/#cluster-mesh
  - https://cilium.io/blog/2019/03/12/clustermesh
  - It isn't a traditional Application Service Mesh ☺
  - Define **globally load balanced services** that span Kubernetes clusters
  - etcd state shared via load-balancers / Nodes communicate over VXLAN / Encryption over IPSec
  - **Selective load balancing to remote clusters is possible but difficult depending on the scenario**
  - Connect to external workloads (e.g., VMs)
- Outcome: It just worked, but it may not be what you need

Microservices Demo Topology

Cluster1



Cluster2

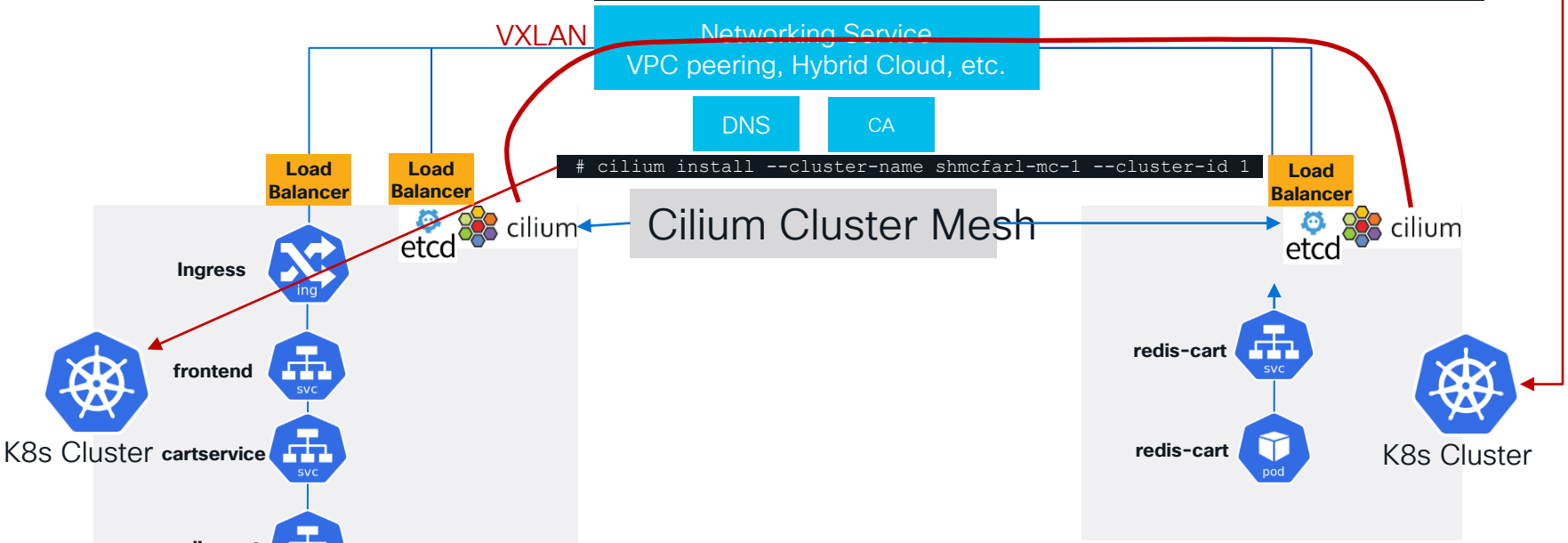


https://github.com/GoogleCloudPlatform/microservices-demo

K8s Multicluster Connectivity – Global Service LB

Setup

```
# cilium install --cluster-name shmcfarl-mc-2 --cluster-id 2 --context
shmcfarl@shmcfarl-mc-2.us-west-2.eksctl.io --inherit-ca shmcfarl@shmcfarl-mc-
1.us-west-2.eksctl.io
```




```
# cilium install --cluster-name shmcfarl-mc-1 --cluster-id 1
```

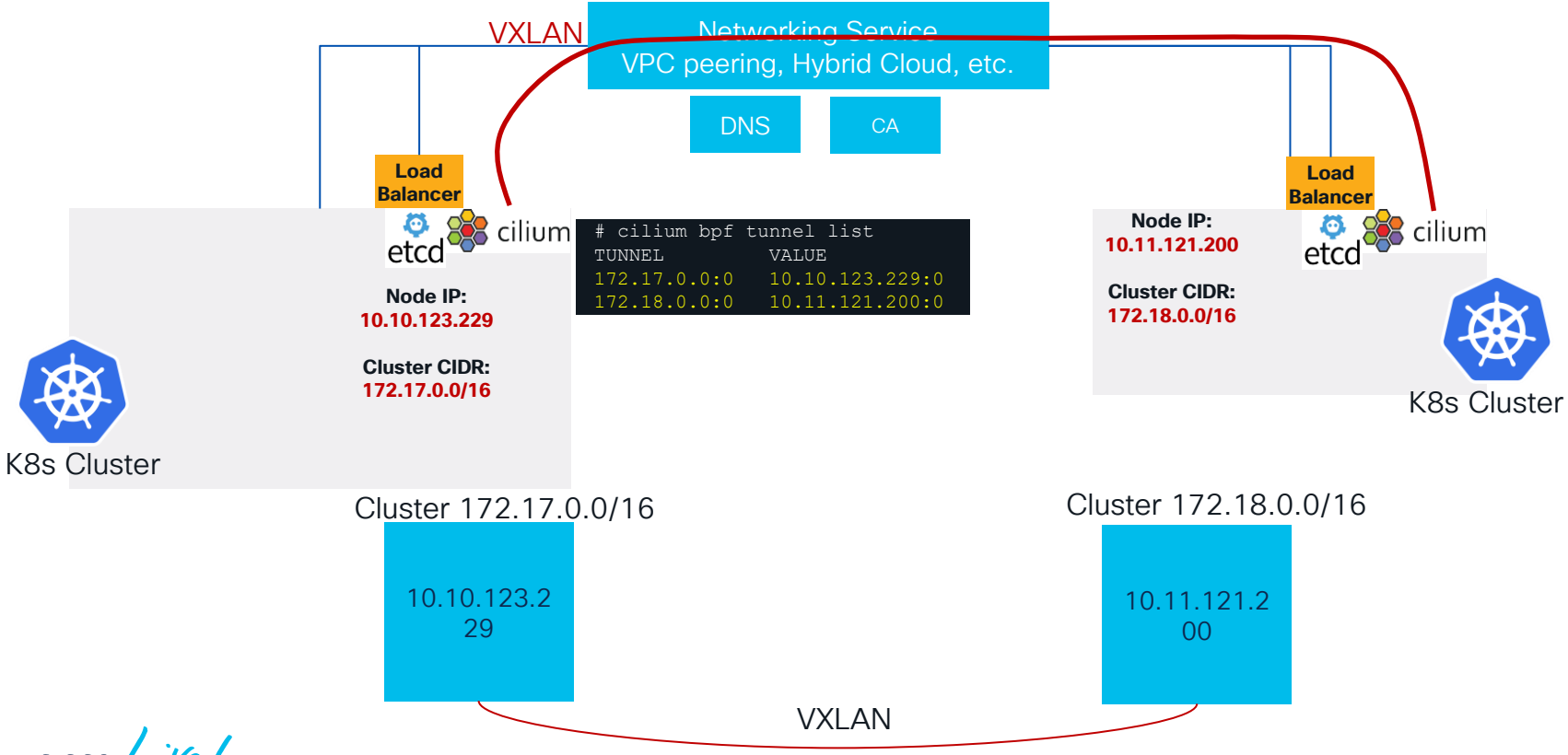
```
# cilium clustermesh enable --context shmcfarl@shmcfarl-mc-1.us-west-2.eksctl.io
# cilium clustermesh enable --context shmcfarl@shmcfarl-mc-2.us-west-2.eksctl.io
```

Control Plane Connectivity: LoadBalancer, NodePort, ClusterIP

```
# cilium clustermesh connect --context shmcfarl@shmcfarl-mc-1.us-west-2.eksctl.io
--destination-context shmcfarl@shmcfarl-mc-2.us-west-2.eksctl.io
```

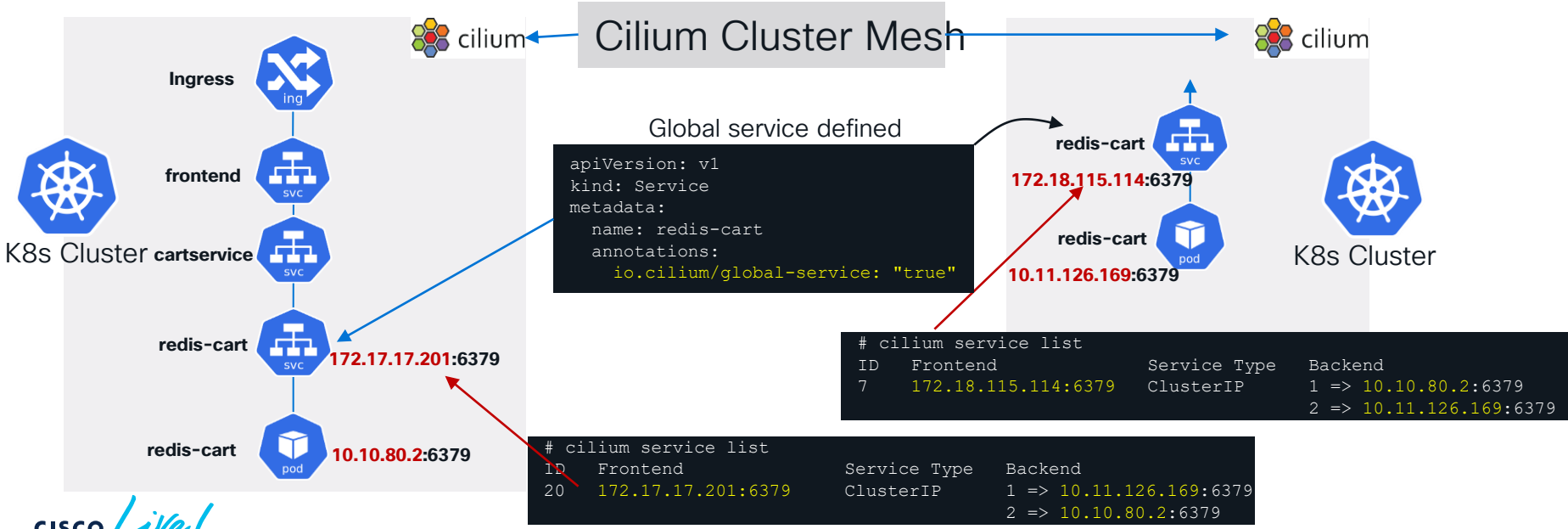
K8s Multicluster Connectivity – Global Service LB

VXLAN

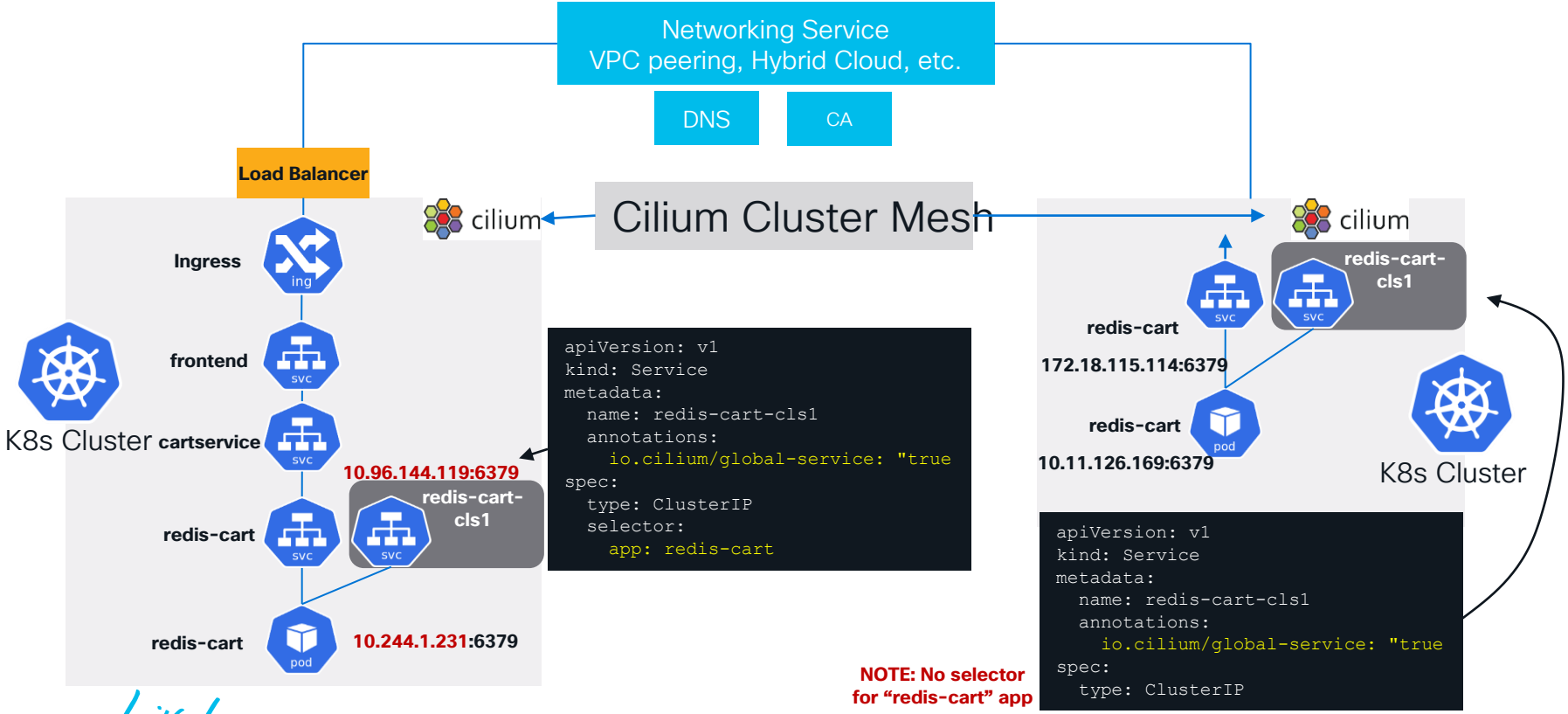


K8s Multicluster Connectivity – Global Service LB

Global LB



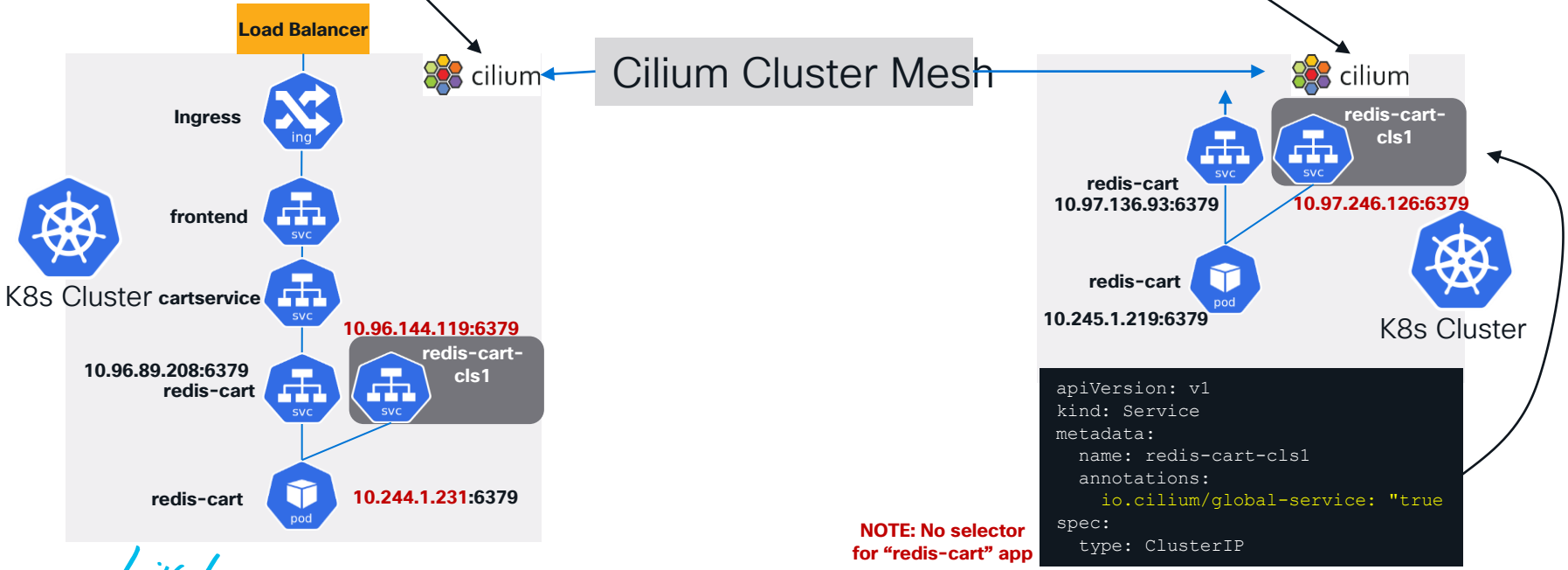
K8s Multicluster Connectivity – Service-to-Service



K8s Multicluster Connectivity – Service-to-Service

```
# cilium service list
ID   Frontend              Service Type   Backend
23   10.96.89.208:6379     ClusterIP      1 => 10.244.1.231:6379
30   10.96.144.119:6379    ClusterIP      1 => 10.244.1.231:6379
```

```
# cilium service list
ID   Frontend              Service Type   Backend
26   10.97.136.93:6379     ClusterIP      1 => 10.245.1.219:6379
32   10.97.246.126:6379    ClusterIP      1 => 10.244.1.231:6379
```



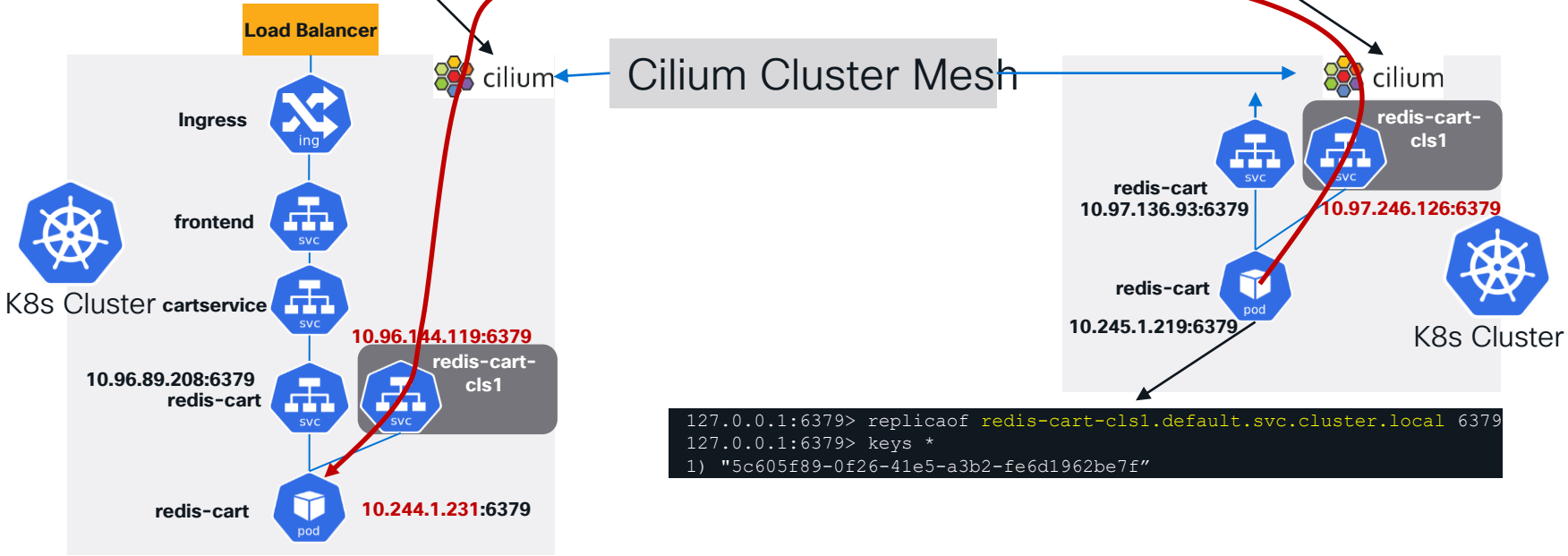

```
apiVersion: v1
kind: Service
metadata:
  name: redis-cart-cls1
  annotations:
    io.cilium/global-service: "true
spec:
  type: ClusterIP
```

NOTE: No selector for "redis-cart" app

K8s Multicluster Connectivity – Service-to-Service

```
# cilium service list
ID   Frontend              Service Type   Backend
23   10.96.89.208:6379     ClusterIP      1 => 10.244.1.231:6379
30   10.96.144.119:6379    ClusterIP      1 => 10.244.1.231:6379
```

```
# cilium service list
ID   Frontend              Service Type   Backend
26   10.97.136.93:6379     ClusterIP      1 => 10.245.1.219:6379
32   10.97.246.126:6379    ClusterIP      1 => 10.244.1.231:6379
```



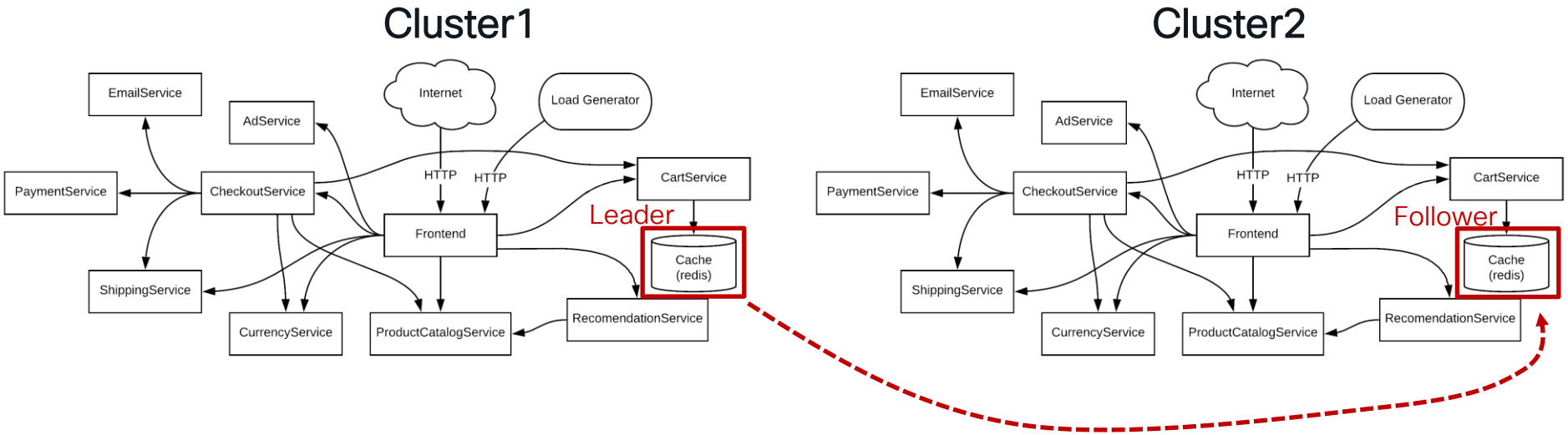

```
127.0.0.1:6379> replicaof redis-cart-cls1.default.svc.cluster.local 6379
127.0.0.1:6379> keys *
1) "5c605f89-0f26-41e5-a3b2-fe6d1962be7f"
```

Submariner

Submariner

- Gateway-based multicluster connectivity for Kubernetes services: https://submariner.io/
- A CNCF project
- What is it?
  - Gateway-based with support for IPSec (libreswan), WireGuard and VXLAN
  - Connect 'exported' services between clusters
  - Can be used as a transport for other stuff like Istio: https://cloud.redhat.com/blog/set-up-istio-multicluster-with-submariner-in-red-hat-advanced-cluster-management-for-kubernetes
- Outcome: It is a very bumpy deployment. Fairly 'smooth' on OpenShift, but bumpy on most other platforms due to out-of-date docs and buggy dependency scripts
- Things to watch out:
  - MTU on pods – Must account for overhead of IPSec/Wireguard/VXLAN
  - Security groups – pay close attention to the SG dependencies per encap type

Microservices Demo Topology

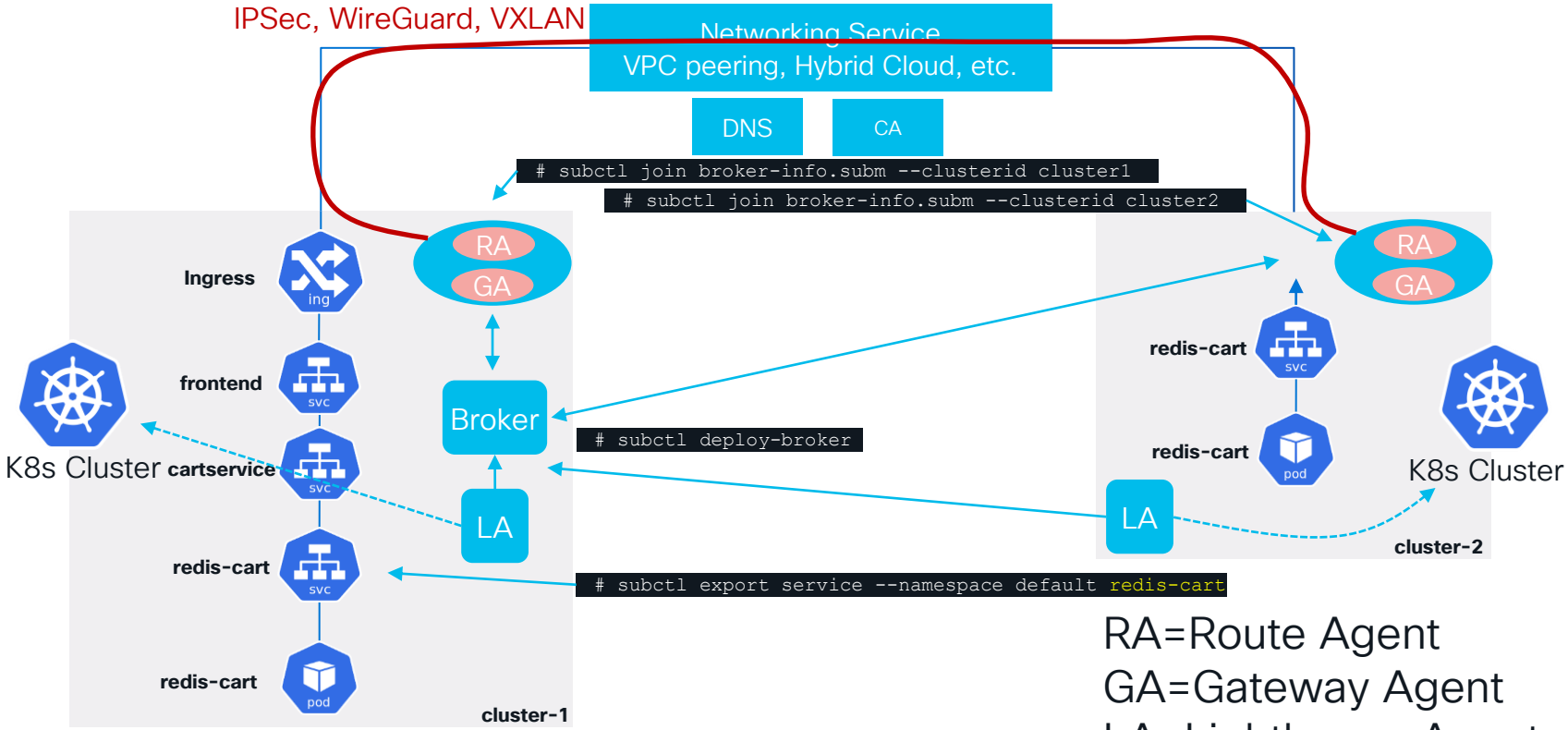


https://github.com/GoogleCloudPlatform/microservices-demo

K8s Multicluster Connectivity – Submariner Service Export

Setup

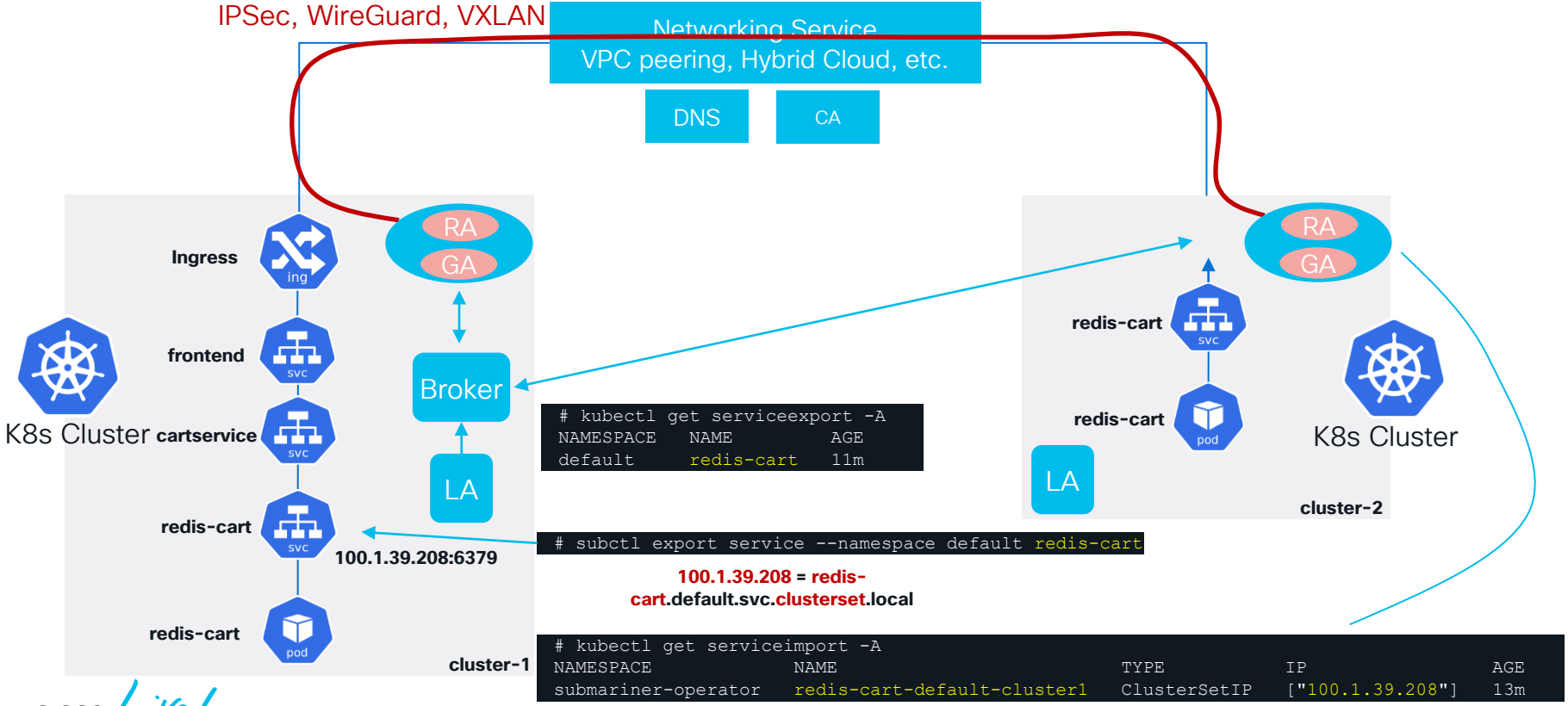
```
# subctl show all
GATEWAY          CLUSTER   REMOTE IP   NAT  CABLE DRIVER  SUBNETS                    STATUS     RTT avg.
cluster2-worker  cluster2  172.18.0.5  no   libreswan     100.2.0.0/16, 10.2.0.0/16  connected  152.062µs
```



RA=Route Agent
GA=Gateway Agent
LA=Lighthouse Agent

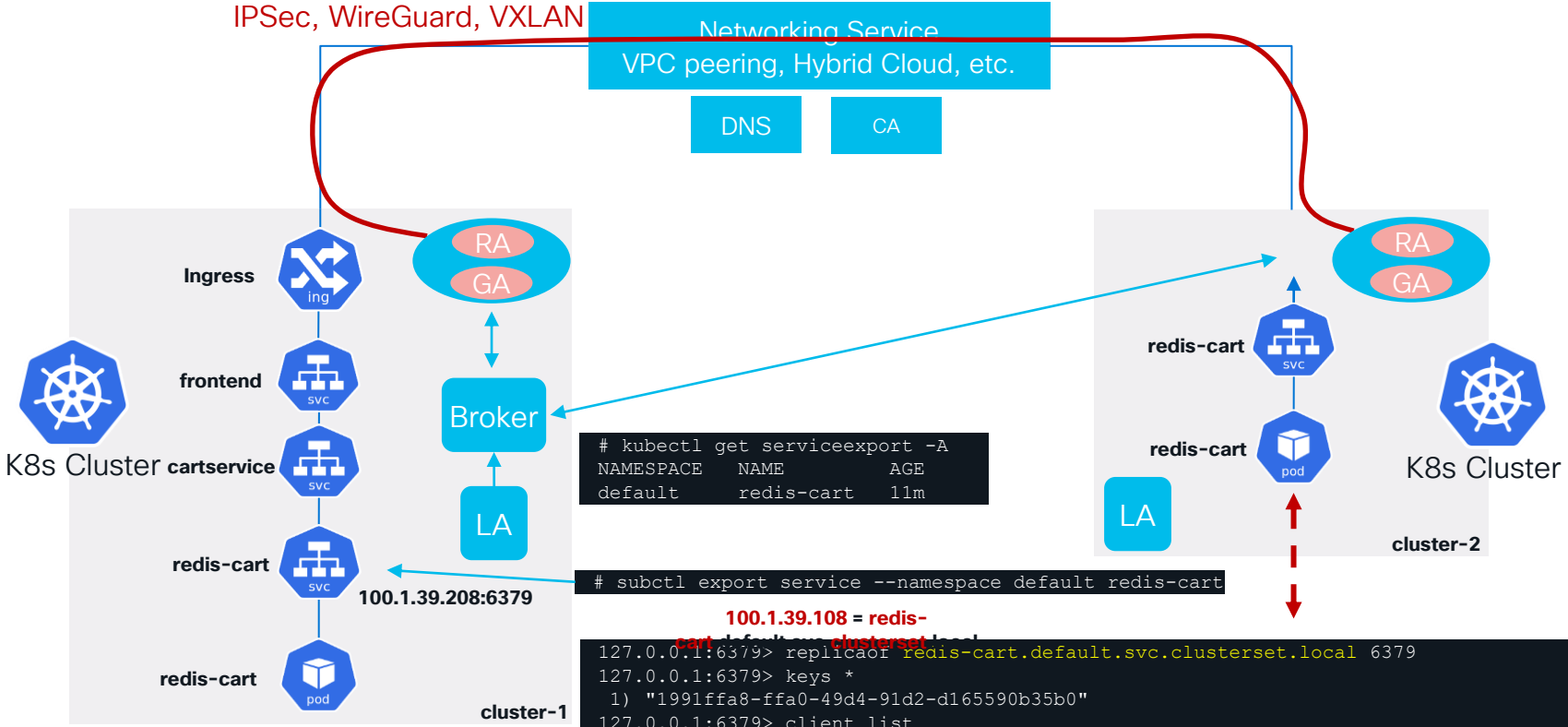
K8s Multicluster Connectivity – Submariner Service Export

Make the Service Known



K8s Multicluster Connectivity – Submariner Service Export

Redis Replication




```
# kubectl get serviceexport -A
NAMESPACE    NAME         AGE
default      redis-cart   11m
```

```
# subctl export service --namespace default redis-cart
```

100.1.39.108 = redis-cart.default.svc.clusterset.local

```
127.0.0.1:6379> replicaof redis-cart.default.svc.clusterset.local 6379
127.0.0.1:6379> keys *
 1) "1991ffa8-ffa0-49d4-91d2-d165590b35b0"
127.0.0.1:6379> client list
. . .
id=29 addr=100.1.39.208:6379
```

Linkerd

Reference

LINKERD

An open source **service mesh** and CNCF project.

- **4 years** in production
- **5,000+** Slack channel members
- **10,000+** GitHub stars
- **100+** contributors

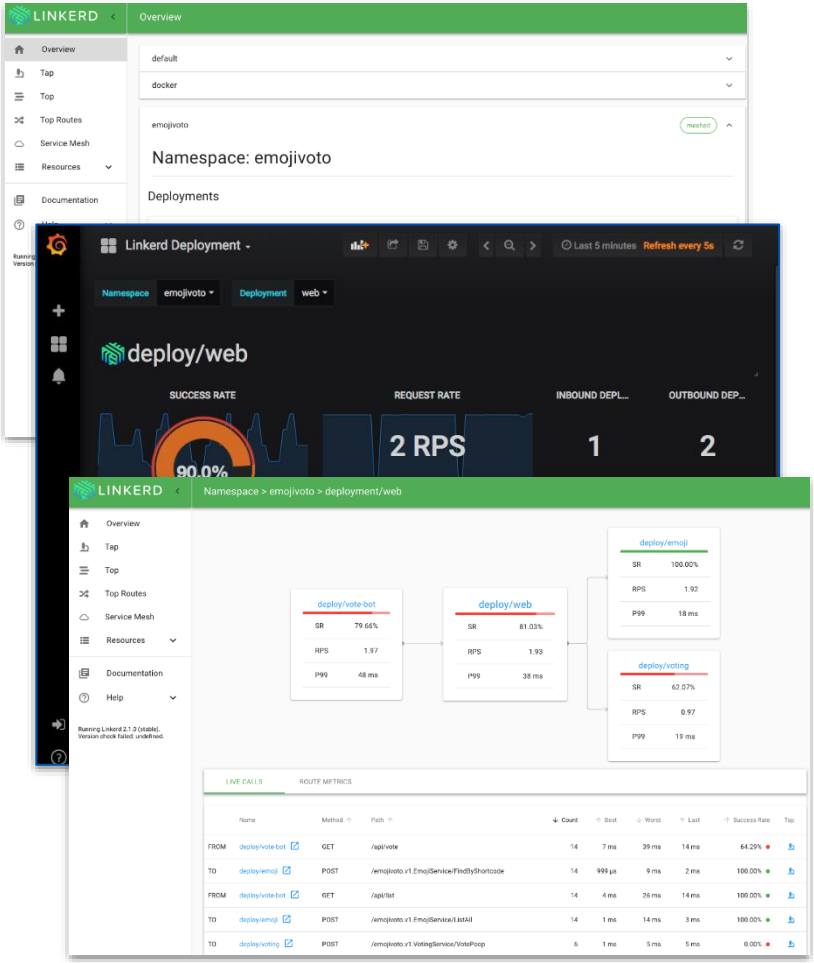
GoDaddy
Walmart
STRAVA
BIGCOMMERCE
Expedia
OfferUp
H-E-B
Cisco webex
Clover Health
planet.

Reference

What does it do?

- Observability: Service-level *golden metrics*: success rates, latencies, throughput. Service topologies.
- Reliability: Retries, timeouts, load balancing, circuit breaking
- Security: Transparent mTLS, cert management and rotation, policy

In an ultralight package focused on operational simplicity first and foremost.



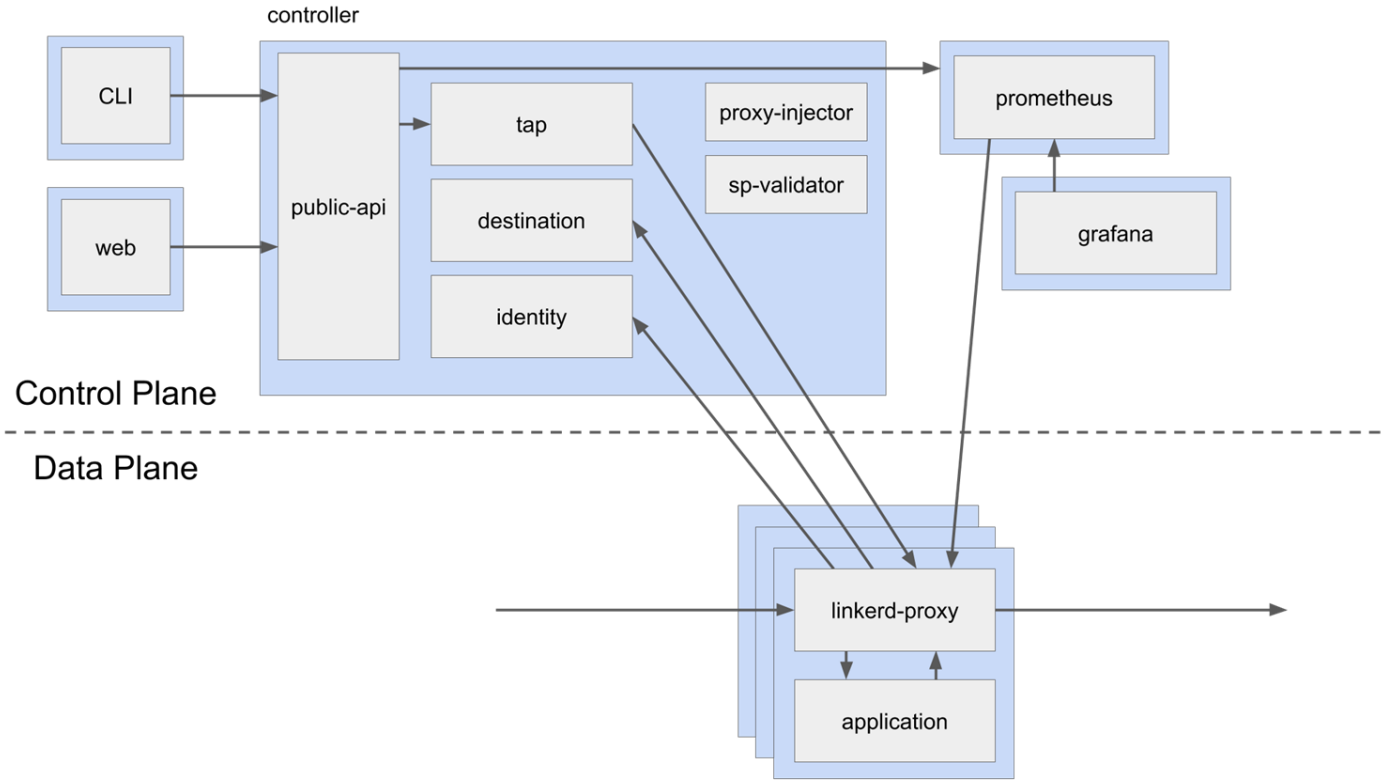
Reference

Linkerd Design

- In short, "do less, not more"
- Just works: Zero config, out of the box, for any Kubernetes app
- Ultralight: Introduce the bare minimum perf and resource cost
- Simple: Reduce operational complexity in every possible way
- Minimal overhead:
  - Control plane: Go. ~200mb RSS (excluding metrics data). (Repo: linkerd/linkerd2).
  - Data plane: Rust. <10mb RSS (Resident Set Size), <1ms p99 (Repo: linkerd/linkerd2-proxy)

Linkerd 2.x Architecture

https://linkerd.io/2/reference/architecture/



Reference

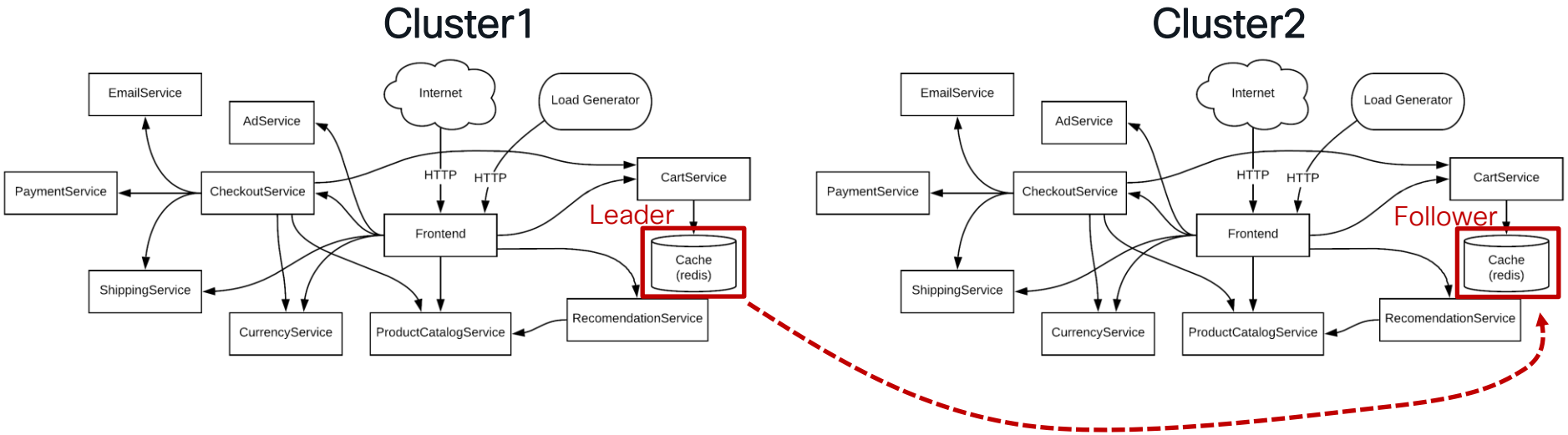
Linkerd: How Do I Get It?

- Where to get it:
  - https://linkerd.io/2/getting-started/
  - Releases: https://github.com/linkerd/linkerd2/releases/
- Deploy a Kubernetes Cluster
- Deploy Linkerd
- Deploy (or add) Linkerd to your microservice(s)

Get involved!

- Linkerd has a friendly, welcoming community! Join us!
- Development is all on https://github.com/linkerd
- Thriving community in the https://slack.linkerd.io/
- Formal announcements on the CNCF https://lists.cncf.io/g/cncf-linkerd-users
- Linkerd is 100% Apache v2 licensed, owned by a neutral foundation (https://www.cncf.io/), and is https://linkerd.io/2019/10/03/linkerds-commitment-to-open-governance/.

Microservices Demo Topology



https://github.com/GoogleCloudPlatform/microservices-demo

Reference

Linkerd Multicluster Setup – Pre-setup Stuff

https://linkerd.io/2.11/getting-started/ https://linkerd.io/2.11/features/multicluster/

- Create certs

```
# step certificate create root.linkerd.cluster.local root.crt root.key \
  --profile root-ca --no-password --insecure

# step certificate create identity.linkerd.cluster.local issuer.crt issuer.key \
  --profile intermediate-ca --not-after 8760h --no-password --insecure \
  --ca root.crt --ca-key root.key
```

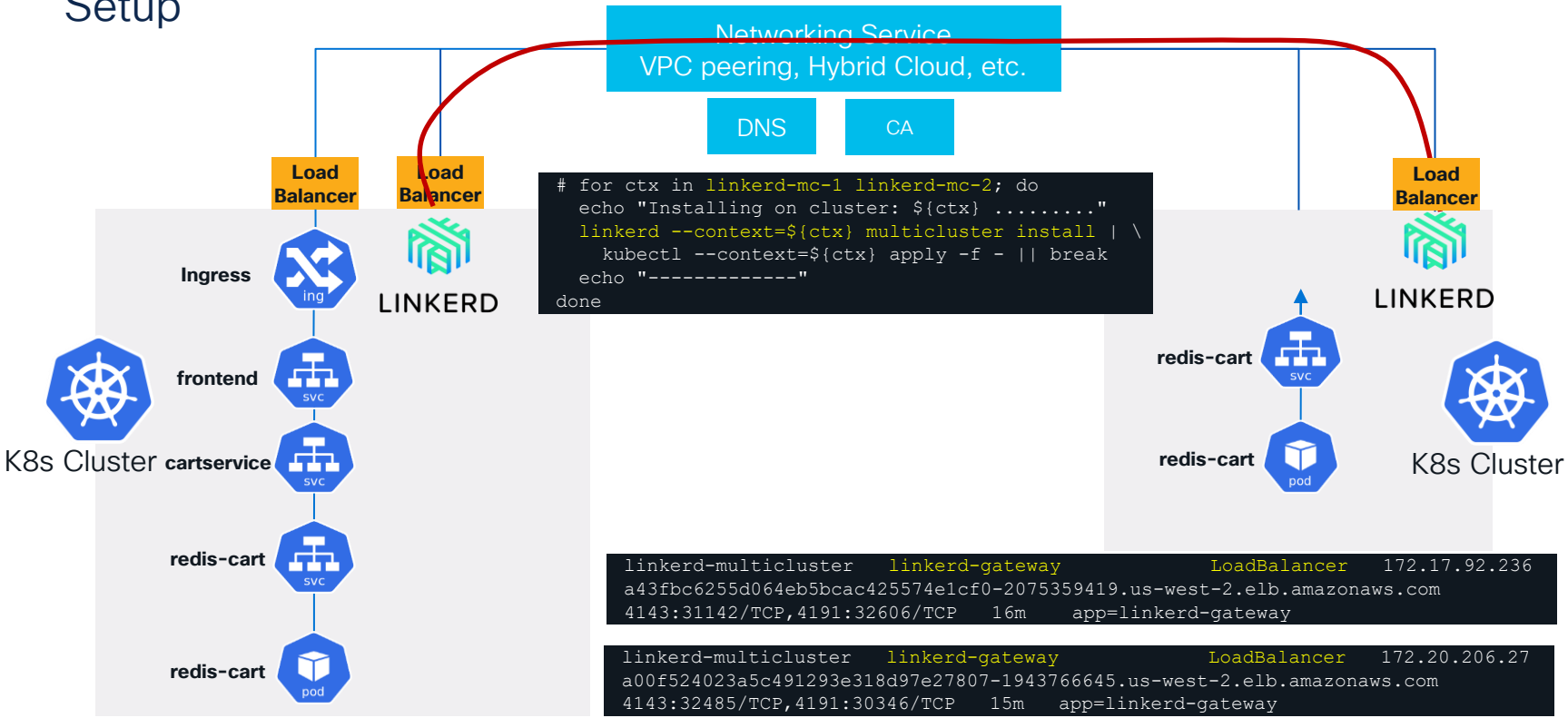
- Install Linkerd and create an anchor of trust between the clusters

```
# linkerd install \
   --identity-trust-anchors-file root.crt \
   --identity-issuer-certificate-file issuer.crt \
   --identity-issuer-key-file issuer.key \
   | tee \
     >(kubectl --context=linkerd-mc-1 apply -f -) \
     >(kubectl --context=linkerd-mc-2 apply -f -)
```

- Follow the documented steps for addons such as Linkerd ‘viz’ (UI) installation

K8s Multicluster Connectivity – Linkerd Multicluster

Setup




```
# for ctx in linkerd-mc-1 linkerd-mc-2; do
  echo "Installing on cluster: ${ctx} ........."
  linkerd --context=${ctx} multicluster install | \
    kubectl --context=${ctx} apply -f - || break
  echo "-------------"
done
```

```
linkerd-multicluster   linkerd-gateway            LoadBalancer   172.17.92.236
a43fbc6255d064eb5bcac425574e1cf0-2075359419.us-west-2.elb.amazonaws.com
4143:31142/TCP,4191:32606/TCP   16m    app=linkerd-gateway
```

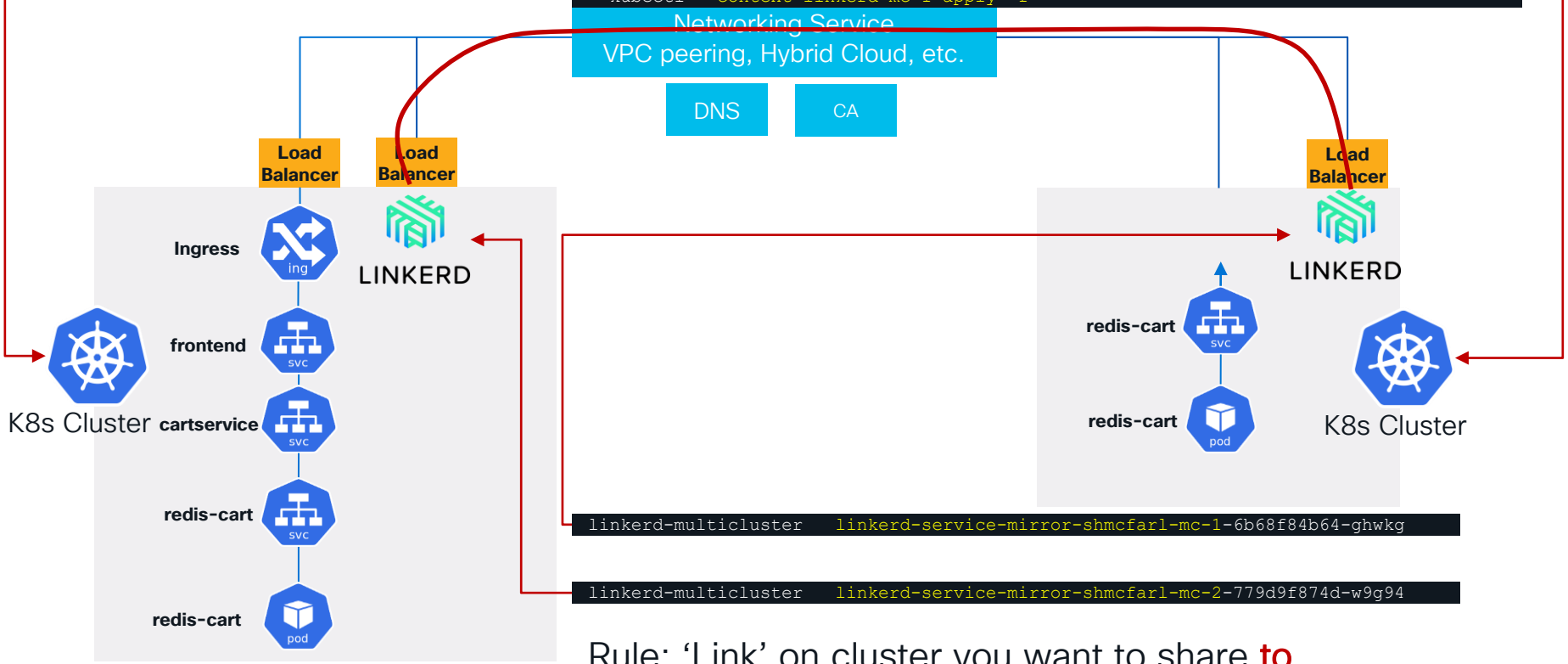
```
linkerd-multicluster   linkerd-gateway            LoadBalancer   172.20.206.27
a00f524023a5c491293e318d97e27807-1943766645.us-west-2.elb.amazonaws.com
4143:32485/TCP,4191:30346/TCP   15m   app=linkerd-gateway
```

K8s Multicluster Connectivity – Linkerd Multicluster

Link Setup

```
# linkerd --context=linkerd-mc-1 multicluster link --cluster-name linkerd-mc-1 |
  kubectl --context=linkerd-mc-2 apply -f -
```

```
# linkerd --context=linkerd-mc-2 multicluster link --cluster-name linkerd-mc-2 |
  kubectl --context=linkerd-mc-1 apply -f -
```



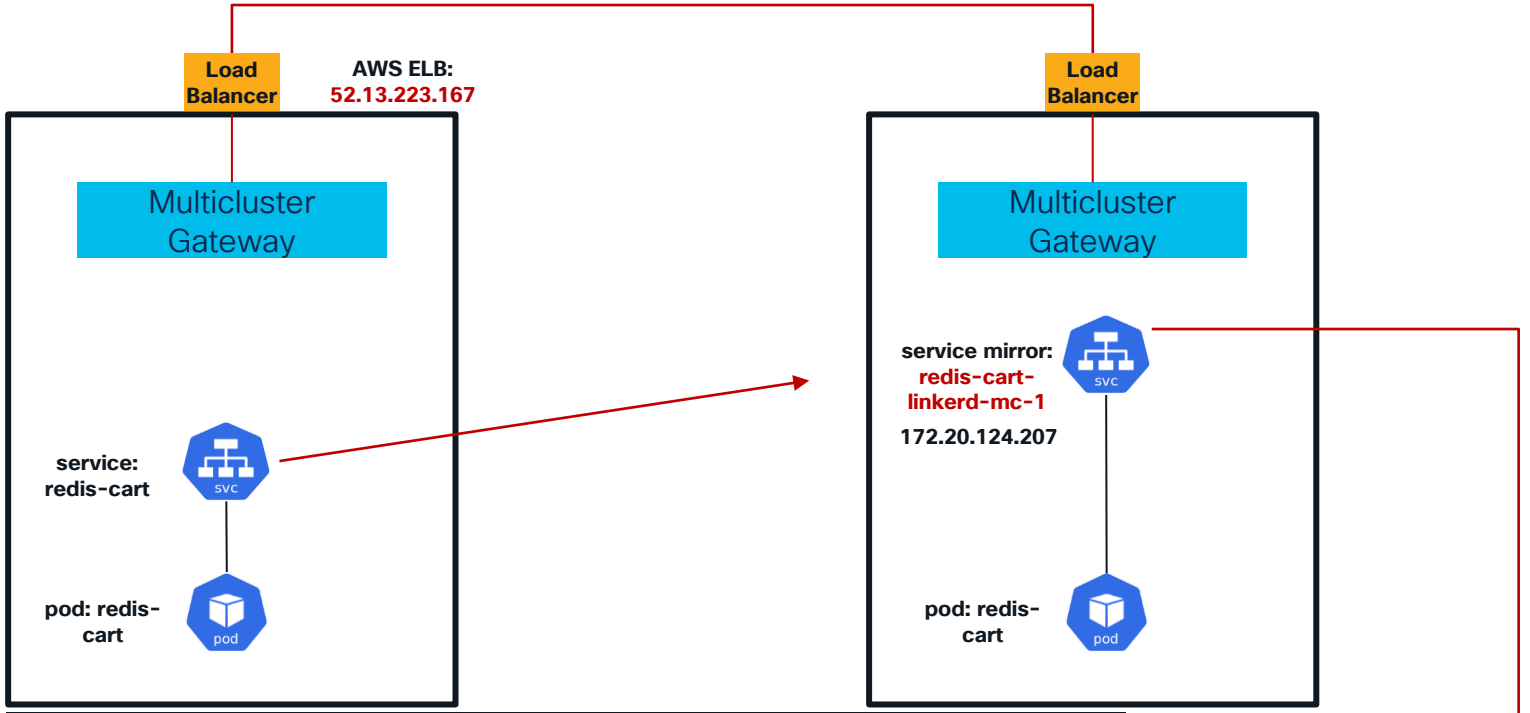

```
linkerd-multicluster   linkerd-service-mirror-shmcfarl-mc-1-6b68f84b64-ghwkg
```

```
linkerd-multicluster   linkerd-service-mirror-shmcfarl-mc-2-779d9f874d-w9g94
```

Rule: ‘Link’ on cluster you want to share **to**

Linkerd Multicluster – Service Export

Rule: 'Link' on cluster you want to share **to**




```
# linkerd --context=linkerd-mc-1 multicluster link --cluster-name linkerd-mc-1 |
  kubectl --context=linkerd-mc-2 apply -f -
```

```
# kubectl --context=linkerd-mc-1 label svc -n default redis-cart mirror.linkerd.io/exported=true
```

```
Name:               redis-cart
Namespace:          default
Labels:             mirror.linkerd.io/exported=true
```

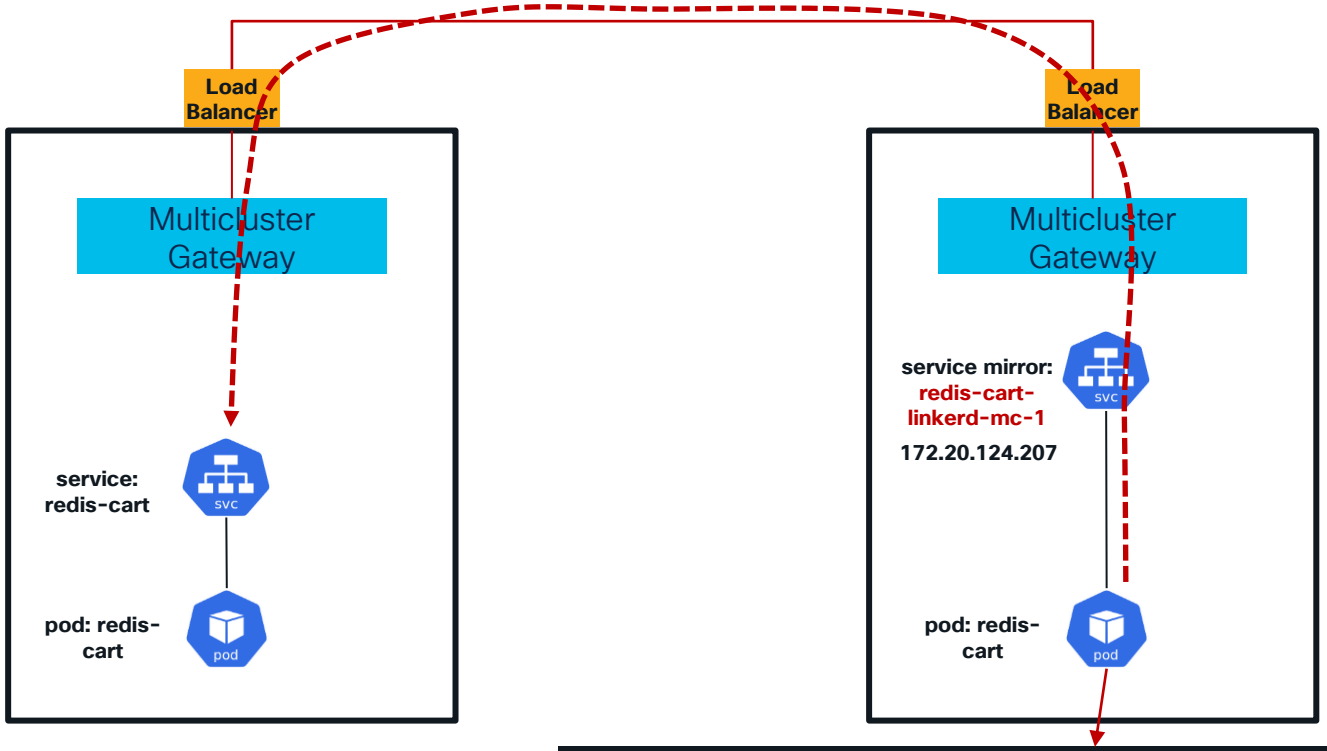
```
# kubectl --context=linkerd-mc-2 get endpoints
NAME                      ENDPOINTS                                  AGE
kubernetes                10.10.154.115:443,10.10.189.103:443        2d3h
redis-cart-linkerd-mc-1   52.13.223.167:4143                         145m
```

Sample of “link” YAML output (highly reduced output)

Reference

```yaml
apiVersion: multicluster.linkerd.io/v1alpha1
kind: Link
metadata:
  name: linkerd-mc-1
  namespace: linkerd-multicluster
spec:
  clusterCredentialsSecret: cluster-credentials-linkerd-mc-1
  gatewayAddress: a9d97fc75ed1d43b19e2a3344ad734cc-1322698043.us-west-2.elb.amazonaws.com
  gatewayIdentity: linkerd-gateway.linkerd-multicluster.serviceaccount.identity.linkerd.cluster.local
  gatewayPort: "4143"
  probeSpec:
    path: /ready
    period: 3s
    port: "4191"
  selector:
    matchExpressions:
    - key: mirror.linkerd.io/exported
      operator: Exists
  targetClusterDomain: cluster.local
  targetClusterLinkerdNamespace: linkerd
  targetClusterName: linkerd-mc-1
---
apiVersion: v1
kind: Service
metadata:
  name: probe-gateway-linkerd-mc-1
  namespace: linkerd-multicluster
  labels:
    mirror.linkerd.io/mirrored-gateway: "true"
    mirror.linkerd.io/cluster-name: linkerd-mc-1
spec:
  ports:
  - name: mc-probe
    port: 4191
    protocol: TCP
```

Linkerd Multicluster – Redis Replication




```
127.0.0.1:6379> replicaof redis-cart-linkerd-mc-1.default.svc.cluster.local 6379
127.0.0.1:6379> keys *
 1) "72c6d371-812f-4778-80c8-c41366d956a6”
```

Istio

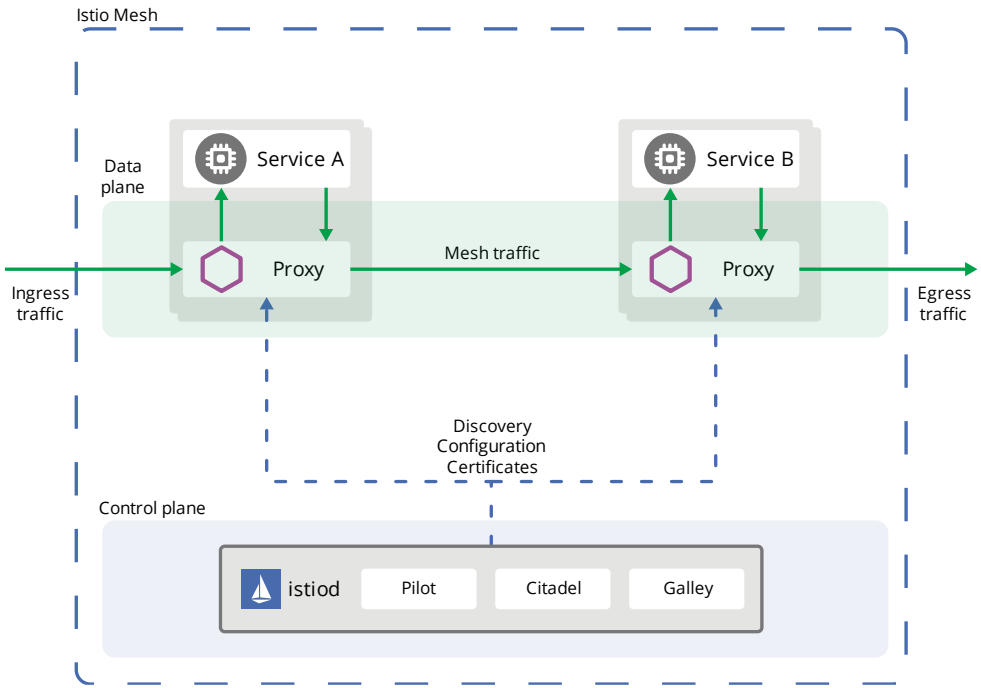
Istio Overview

- 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/
- https://istio.io/docs/concepts/what-is-istio/
  - 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

Istio Architecture

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

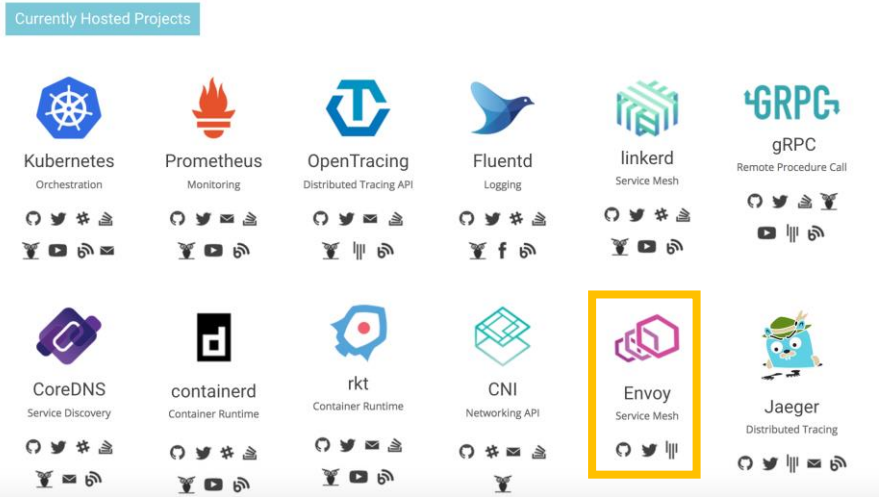
- istiod
  - Pilot
    - Handles service discovery and config data
    - Provides the Envoy proxies with the mesh topology and route rules
  - Galley
    - Validates user authored Istio API configuration on behalf of other control plane components
    - Top-level config ingestion, processing and distribution
  - 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



Envoy

https://www.envoyproxy.io

- Implemented by Lyft
- A C++ based L4/L7 proxy
- Can be used independently of any service mesh (Istio)



- API driven
- Traffic routing and splitting
- Transparent proxying
- Health checks, circuit breakers, etc.

USED BY

https://github.com/envoyproxy/envoy

Istio: How Do I Get It?

- Where to get it:
  - Istio currently is available directly from the Istio community at: https://istio.io/about/community/join/
  - It can also be built directly: https://github.com/istio/istio
  - It can be enabled as an infrastructure option in GKE
- How to install it (Kubernetes):
  - https://istio.io/docs/setup/getting-started/
  - Kubernetes installation is a prerequisite
  - Directly from the manifests included in the release
  - Using Helm charts included in the release

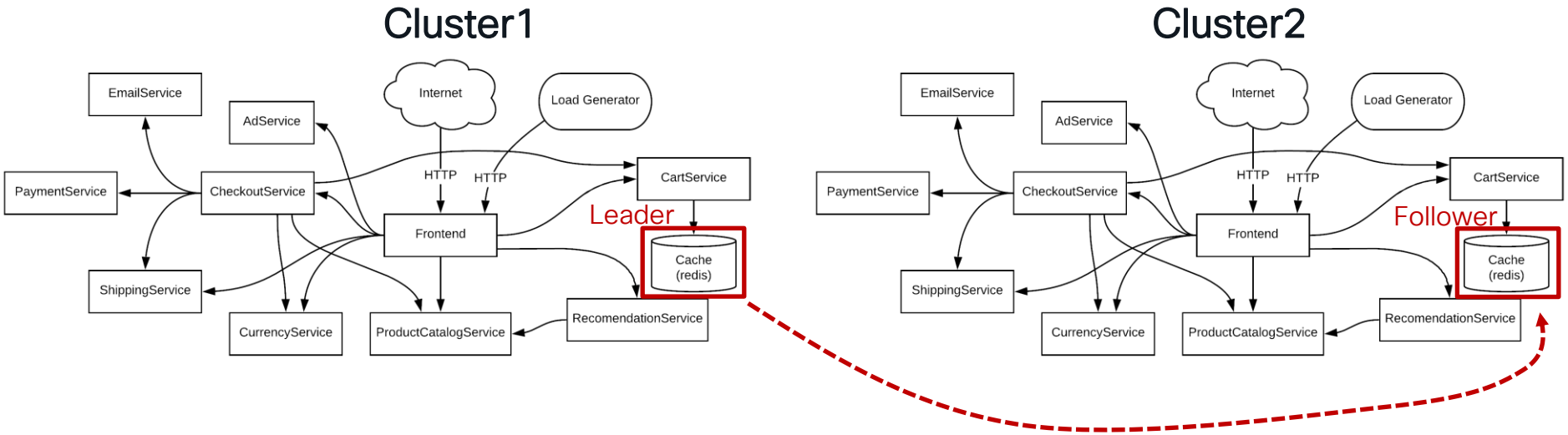
Contribution

- Contribution Readme: https://github.com/istio/community/blob/master/CONTRIBUTING.md
- Contributing to the Docs: https://istio.io/about/contribute/
- Istio Discussion: https://discuss.istio.io/

Istio Multicluster

- 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 network” ->> “flat networking”, “shared networking” = full reachability between workloads without an Istio gateway
- “multiple network” – Workloads reach each other via an Istio gateway
- Pre-planning:
  - Service naming
  - Istio DNS proxy
  - Service sharing/exposure – Control at the gateway or via Istio Authorization: https://istio.io/latest/docs/reference/config/security/authorization-policy/

Microservices Demo Topology



https://github.com/GoogleCloudPlatform/microservices-demo

Reference

Istio Multicluster Setup – Pre-setup Stuff

https://istio.io/latest/docs/setup/install/multicluster/before-you-begin/

- Create certs and secrets on first cluster and 2nd clusters

```
# make -f ../tools/certs/Makefile.selfsigned.mk root-ca
# make -f ../tools/certs/Makefile.selfsigned.mk cluster1-cacerts
# kubectl create namespace istio-system
# kubectl create secret generic cacerts -n istio-system \
        --from-file=cluster1/ca-cert.pem \
        --from-file=cluster1/ca-key.pem \
        --from-file=cluster1/root-cert.pem \
        --from-file=cluster1/cert-chain.pem
```

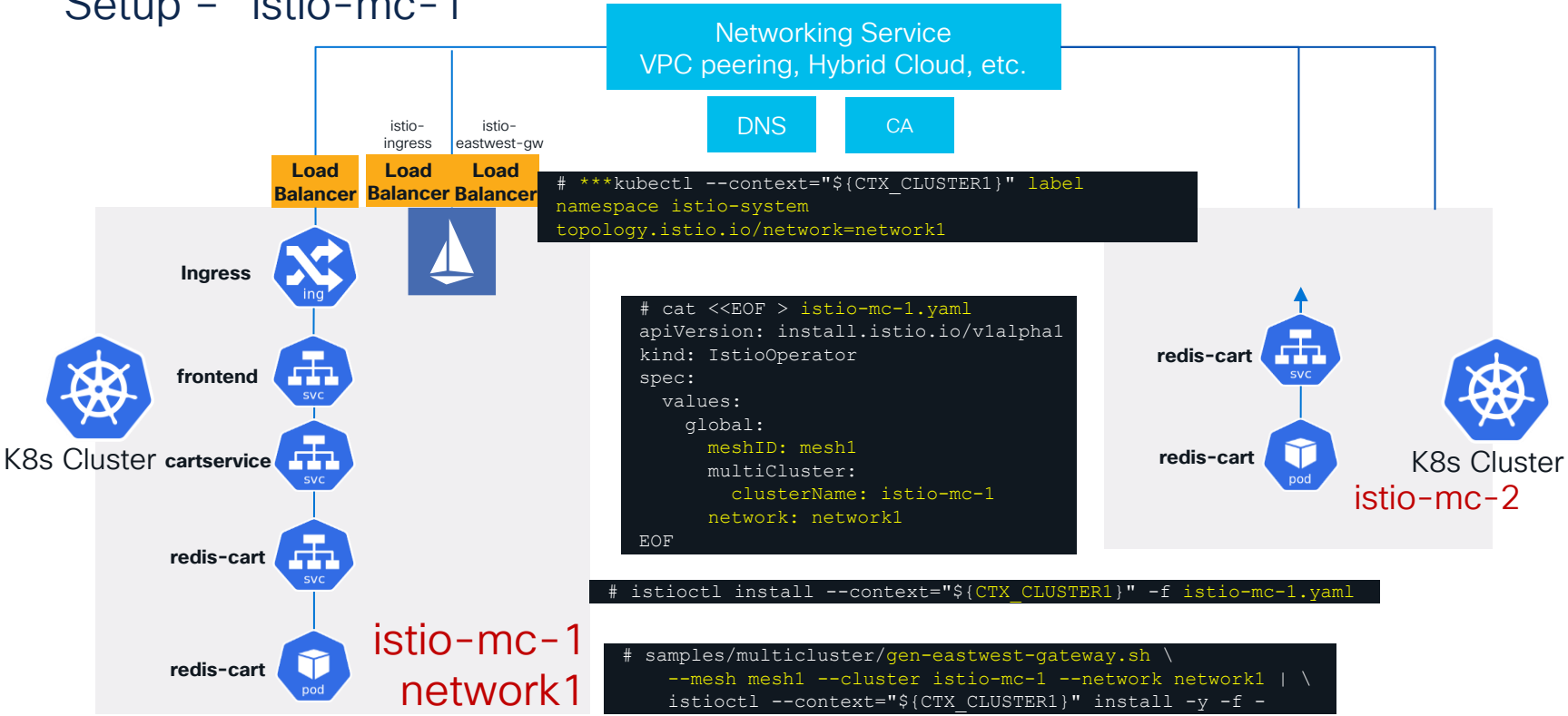
```
# make -f ../tools/certs/Makefile.selfsigned.mk cluster2-cacerts
# kubectl create namespace istio-system
# kubectl create secret generic cacerts -n istio-system \
        --from-file=cluster2/ca-cert.pem \
        --from-file=cluster2/ca-key.pem \
        --from-file=cluster2/root-cert.pem \
        --from-file=cluster2/cert-chain.pem
```

- Export context info for future use with ‘kubectl’ and ‘istioctl’

```
# export CTX_CLUSTER1=istio-mc-1
# export CTX_CLUSTER2=istio-mc-2
```

K8s Multicluster Connectivity – Istio Multicluster

Setup – “istio-mc-1”




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

```
# cat <<EOF > istio-mc-1.yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    global:
      meshID: mesh1
      multiCluster:
        clusterName: istio-mc-1
      network: network1
EOF
```

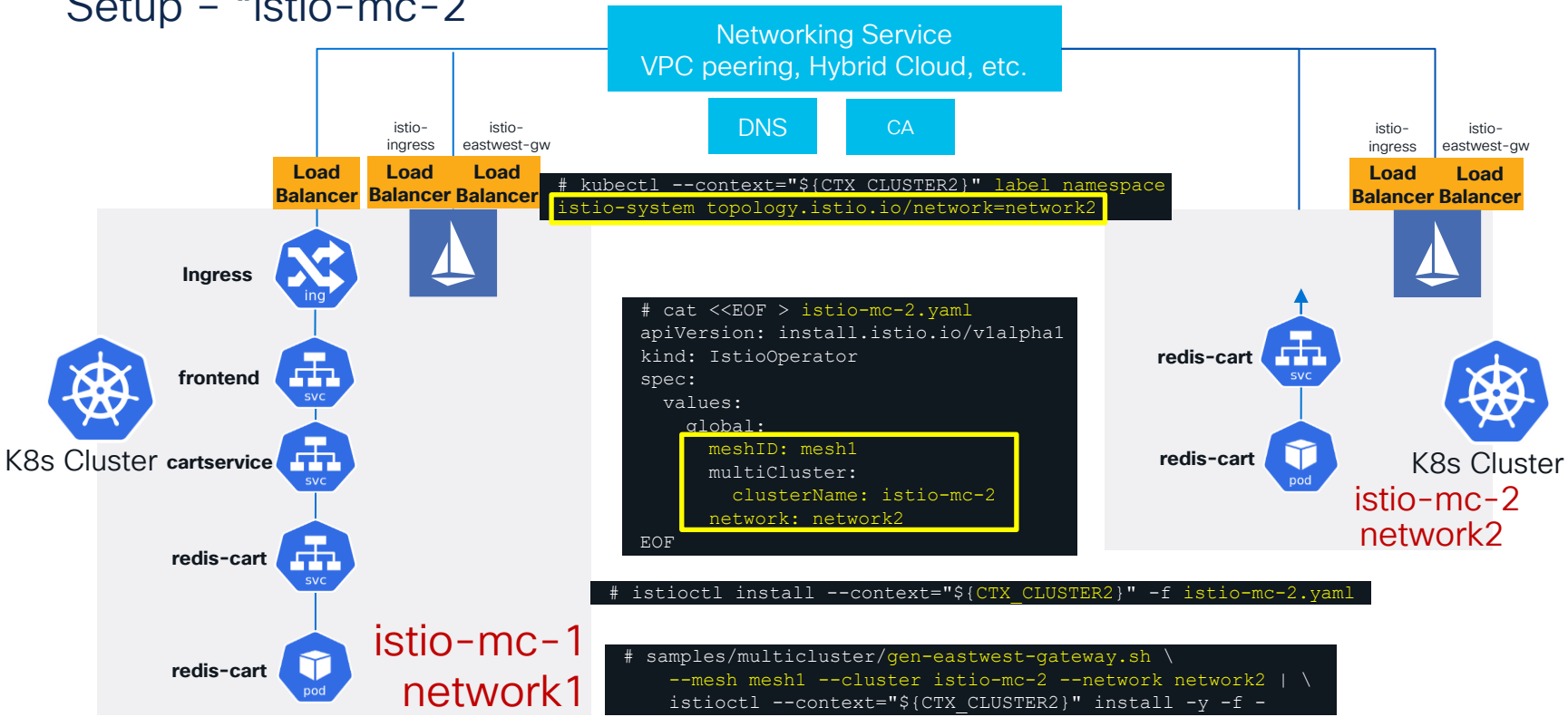
```
# istioctl install --context="${CTX_CLUSTER1}" -f istio-mc-1.yaml
```

```
# samples/multicluster/gen-eastwest-gateway.sh \
    --mesh mesh1 --cluster istio-mc-1 --network network1 | \
    istioctl --context="${CTX_CLUSTER1}" install -y -f -
```

***Do this on existing Istio deployment

K8s Multicluster Connectivity – Istio Multicluster

Setup – “istio-mc-2”




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

```
# cat <<EOF > istio-mc-2.yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  values:
    global:
      meshID: mesh1
      multiCluster:
        clusterName: istio-mc-2
      network: network2
EOF
```

```
# istioctl install --context="${CTX_CLUSTER2}" -f istio-mc-2.yaml
```

```
# samples/multicluster/gen-eastwest-gateway.sh \
    --mesh mesh1 --cluster istio-mc-2 --network network2 | \
    istioctl --context="${CTX_CLUSTER2}" install -y -f -
```

Warning: It is ALWAYS DNS that kills you ☺

- 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, "**redis-cart-cls1.default.svc.cluster.local**" will not be resolvable on the 2nd cluster

Add to the Istio Operator Config

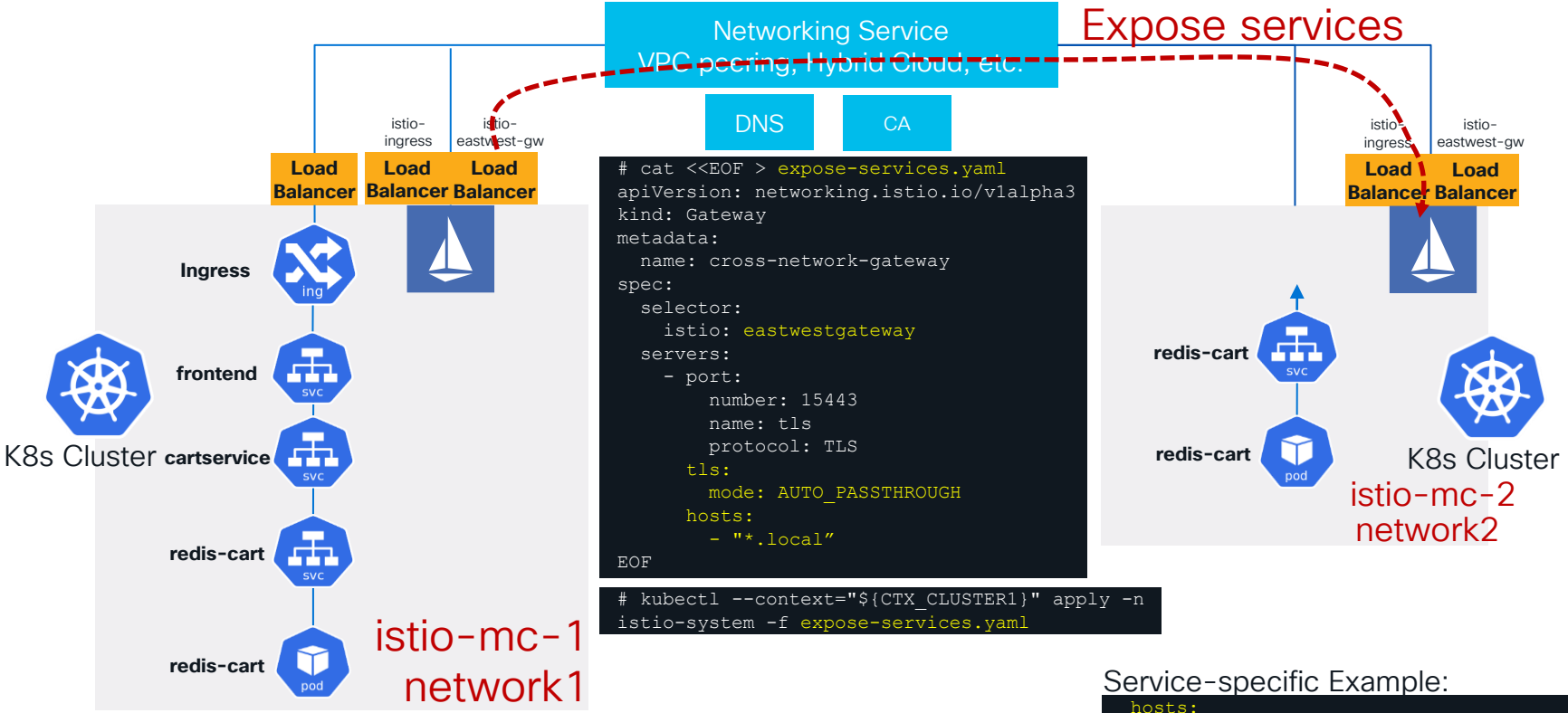
```
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    defaultConfig:
      proxyMetadata:
        # Enable basic DNS proxying
        ISTIO_META_DNS_CAPTURE: "true"
```

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"
```

K8s Multicluster Connectivity – Istio Multicluster

Expose Services – “istio-mc-1”




```
# cat <<EOF > expose-services.yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: cross-network-gateway
spec:
  selector:
    istio: eastwestgateway
  servers:
    - port:
        number: 15443
        name: tls
        protocol: TLS
      tls:
        mode: AUTO_PASSTHROUGH
      hosts:
        - "*.local”
EOF
```

```
# kubectl --context="${CTX_CLUSTER1}" apply -n
istio-system -f expose-services.yaml
```

Service-specific Example:

```
  hosts:
        - ”<SVC>.default.svc.cluster.local”
```

Reference

Istio Multicluster Setup – Endpoint Discovery

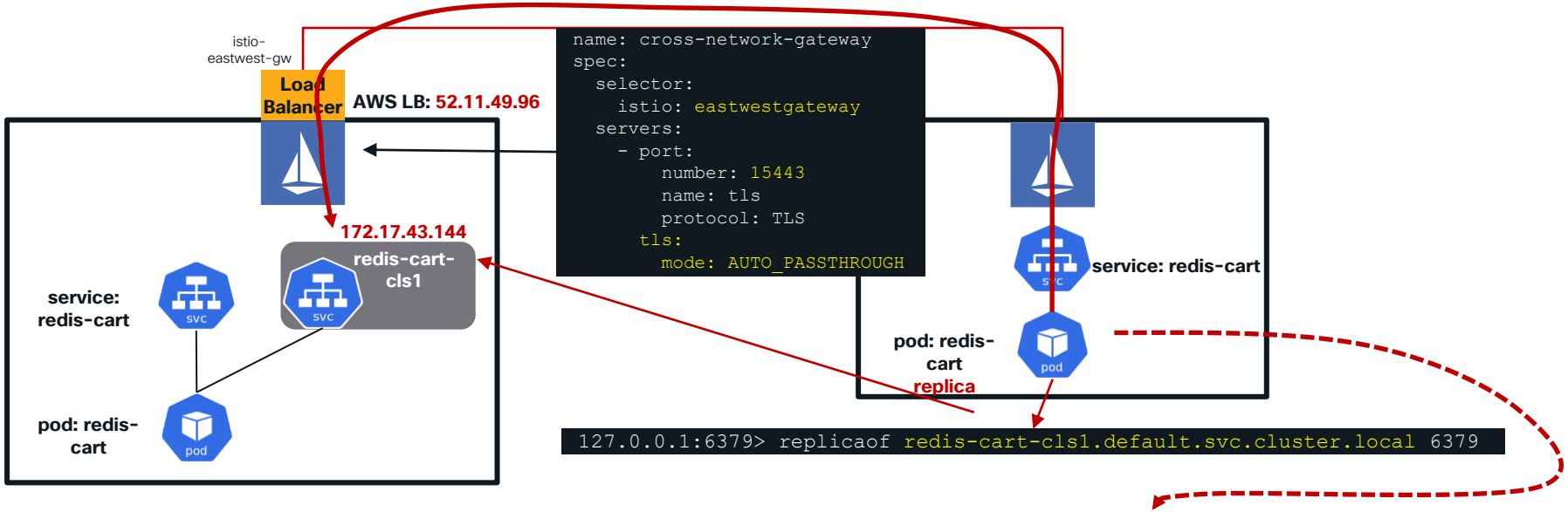
- Install remote secrets in both clusters so that each cluster has API server access to the other cluster

```
# istioctl x create-remote-secret \
  --context="${CTX_CLUSTER1}" \
  --name=istio-mc-1 | \
  kubectl apply -f - --context="${CTX_CLUSTER2}"
```

```
# istioctl x create-remote-secret \
  --context="${CTX_CLUSTER2}" \
  --name=istio-mc-2 | \
  kubectl apply -f - --context="${CTX_CLUSTER1}"
```

Istio Multicluster – Redis Replication

Service Mirror – Phantom/Ghost Services




```
# istioctl proxy-config endpoints --context $CTX_CLUSTER2 redis-cart-5b569cd47-6ppzm --cluster
"outbound|6379||redis-cart-cls1.default.svc.cluster.local"
ENDPOINT                    STATUS      OUTLIER CHECK     CLUSTER
52.11.49.96:15443           HEALTHY     OK                outbound|6379||redis-cart-cls1.default.svc.cluster.local
```

```
istioctl proxy-config listeners --context $CTX_CLUSTER2 redis-cart-5b569cd47-6ppzm --port 6379 -o json
. . .
        "name": "172.17.43.144_6379",
. . .
"type.googleapis.com/envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy",
                            "statPrefix": "outbound|6379||redis-cart-cls1.default.svc.cluster.local",
                            "cluster": "outbound|6379||redis-cart-cls1.default.svc.cluster.local"
```

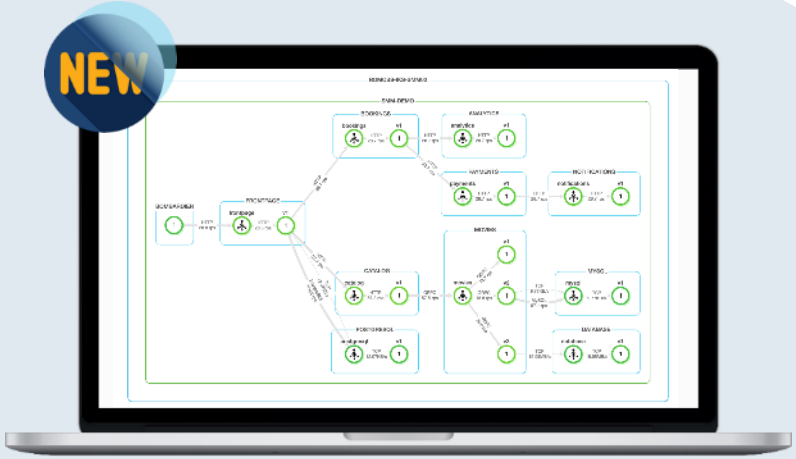
Example of redis-cart-cls1 service

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Service
metadata:
  name: redis-cart-cls1
spec:
  type: ClusterIP
  selector:
    app: redis-cart
  ports:
  - name: tcp-redis
    protocol: TCP
    port: 6379
    targetPort: 6379
EOF
```

Cisco Calisti – A Service Mesh Manager

Cisco Calisti

https://calisti.app/



Introducing extensions
for Intersight Kubernetes Service

Operationalize the service mesh

Multi-cloud, multi-cluster connectivity and observability
Connect any on-prem and public cloud together

Simplifies service mesh management
Single pane of glass, in depth metrics

Policy-based app networking & security
Policy management for DevOps teams

Traffic management ensures smooth app updates

Complete application and health **observability**

Security at all layers between clusters and clouds

Cisco Calisti Benefits

1

Multi-Cluster Observability

- ✓ Proactive issue resolution using SLO[1], error budgeting, actionable alerting when SLO's are endangered
- ✓ Faster root cause resolution using timeline view, outlier detection, traffic tapping/tracing
- ✓ Better visibility into service-to-service performance through Traffic Analytics

2

Simplified mesh & traffic management

- ✓ Complete Istio lifecycle mgmt.
- ✓ Ensure High Availability via automated tooling, metrics
- ✓ Rich, comprehensive operations focused dashboard
- ✓ Enterprise-grade security hardening & lifecycle
- ✓ Reduced risk of day 2 deployments via canary upgrades
- ✓ Reduce human error via config validation
- ✓ VM-extensions for brownfield and external service linkage

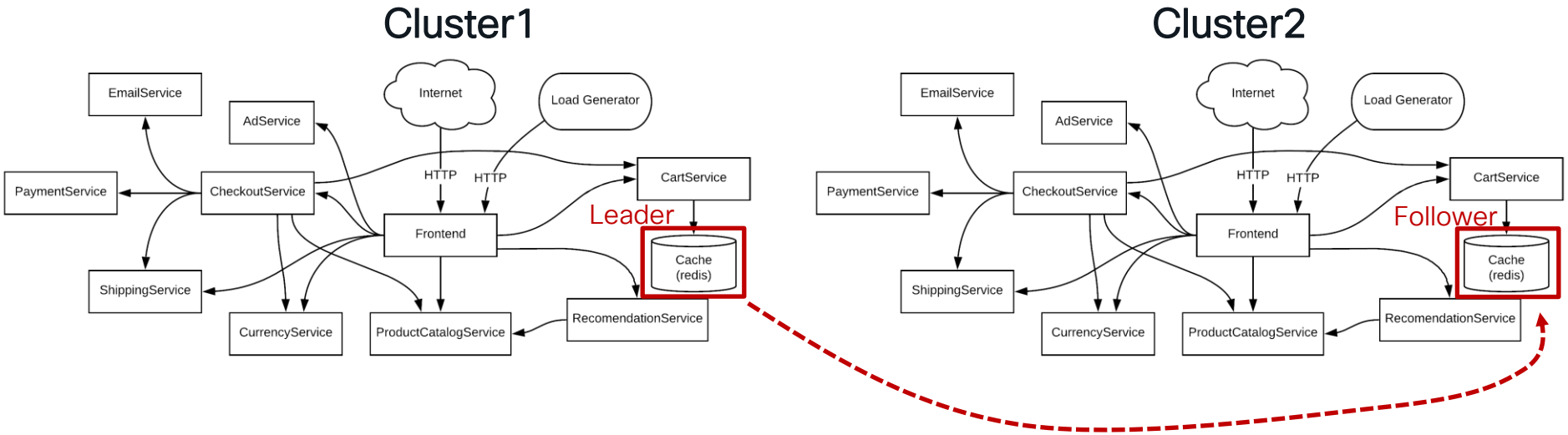
3

Policy based n/w & Security

- ✓ Simplified application deployment via security, observability and platform traffic management
- ✓ Respond quickly to security vulnerabilities via policy enforcement
- ✓ Avoid issues via canary deployments, circuit breakers
- ✓ DevOps friendly traffic debugging

[1]Service Level Objectives

Microservices Demo Topology



https://github.com/GoogleCloudPlatform/microservices-demo

Cisco Calisti Setup – From 19 steps to 3 ☺

https://calisti.app/

1) Install Cisco Calisti and identify the first cluster name:

```
# smm install -a --cluster-name smm-mc-1
```

2) Install Cisco Calisti with a full Istio control plane and attach the 2nd cluster to the 1st cluster:

```
# smm istio cluster attach smm-mc-2.yaml --active-istio-control-plane
```

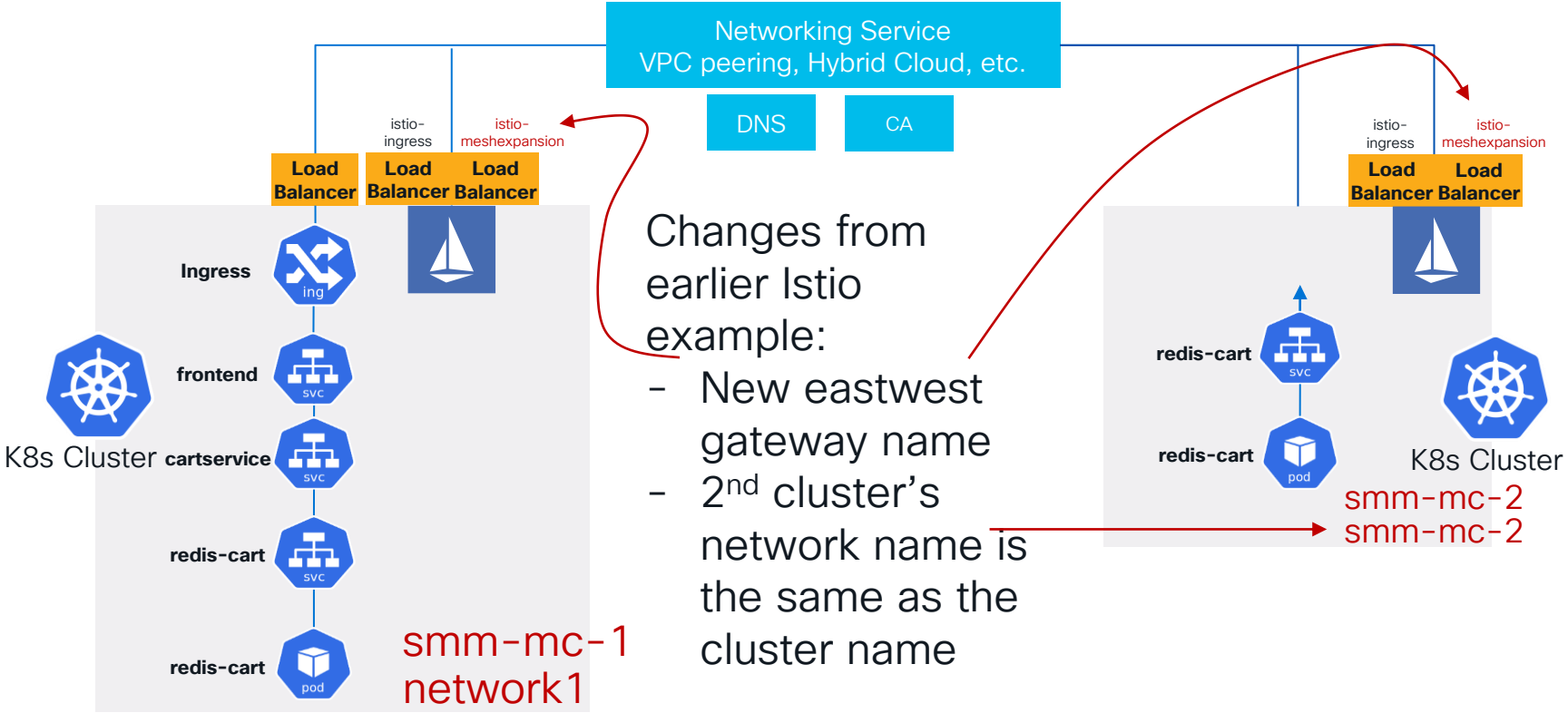
3) Enable Istio sidecar injection on a namespace:

```
# smm sidecar-proxy auto-inject on default
```

```
# smm istio cluster status
Clusters
---
Name       Type   Provider  Regions      Version                    Distribution  Status  Message
smm-mc-1   Local  amazon    [us-east-2]  v1.21.2-13+d2965f0db10712  EKS           Ready
smm-mc-2   Peer   amazon    [us-east-2]  v1.21.2-13+d2965f0db10712  EKS           Ready
ControlPlanes
---
Cluster    Name                     Version  Trust Domain     Pods
Proxies
smm-mc-1   cp-v111x.istio-system  1.11.4   [cluster.local]  [istiod-cp-v111x-75b7ccbb76-6szk9.istio-system]  32/32
smm-mc-2   cp-v111x.istio-system  1.11.4   [cluster.local]  [istiod-cp-v111x-6f5d85c56f-vw2k7.istio-system]  5/5
```

Demo

K8s Multicluster Connectivity – Cisco Calisti Multicluster



Changes from earlier Istio example:

- New eastwest gateway name
- 2nd cluster's network name is the same as the cluster name

Reference

2nd Warning: It is ALWAYS DNS that kills you ☺

- 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, "**redis-cart-cls1.default.svc.cluster.local**" will 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"
```

K8s Multicluster Connectivity – Cisco Calisti Multicluster

- Brute force – Control which services are exposed/shared at the gateway

```
# kubectl edit -n istio-system gw istio-cross-network-cp-v111x
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: istio-cross-network-cp-v111x
spec:
  servers:
  - hosts:
    - "*.local”
```

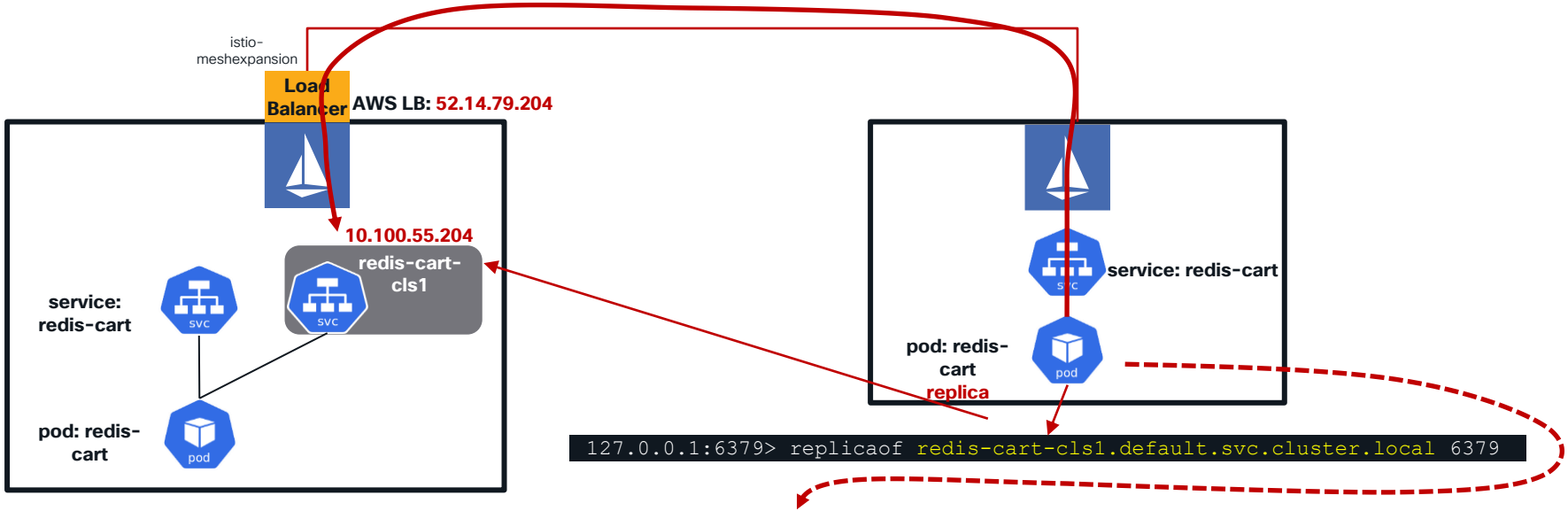
Service-specific Example:

```
 hosts:
        - ”<SVC>.default.svc.cluster.local”
```

- Istio Authorization Policy: https://istio.io/latest/docs/reference/config/security/authorization-policy/
  - Microscopic control of which things talk to which other things and how

Cisco Calisti Multicluster – Redis Replication

Service Mirror – Phantom/Ghost Services




```
# istioctl proxy-config endpoints redis-cart-5b569cd47-brxgr --cluster "outbound|6379||redis-cart-cls1.default.svc.cluster.local"
ENDPOINT                    STATUS      OUTLIER CHECK     CLUSTER
52.14.79.204:15443           HEALTHY      OK                outbound|6379||redis-cart-cls1.default.svc.cluster.local
```

```
# istioctl proxy-config listeners redis-cart-5b569cd47-brxgr --port 6379 -o json
. . .
        "name": "10.100.55.204_6379",
. . .
                                "type.googleapis.com/envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy",
                                "statPrefix": "outbound|6379||redis-cart-cls1.default.svc.cluster.local",
                                "cluster": "outbound|6379||redis-cart-cls1.default.svc.cluster.local"
```

Reference

Cisco Calisti – Multicluster – Multi-Control Plane

Cisco SMM | MESH

CONTROL PLANES	CLUSTERS	ISTIO PROXIES MEMORY USAGE	ISTIO PROXIES CPU USAGE	ISTIO PROXIES NOT RUNNING
2	2	2.33GB	0.29vCPU	0

Clusters

NAME	TYPE	PROVIDER	VERSION	STATUS
smm-mc-1	Local	amazon (us-east-2)	v1.21.2-13+d2965f0db10712 (EKS)	Ready
smm-mc-2	Peer	amazon (us-east-2)	v1.21.2-13+d2965f0db10712 (EKS)	Ready

Control planes

NAME	CLUSTER	VERSION	TRUST DOMAIN	PODS	PROXIES	CONFIG
cp-v111x.istio-system	smm-mc-1	1.11.4	cluster.local	istiod-cp-v111x-75b7ccbb76-6szk9.istio-system	32 / 32	
cp-v111x.istio-system	smm-mc-2	1.11.4	cluster.local	istiod-cp-v111x-6f5d85c56f-vw2k7.istio-system	5 / 5	

Cisco Calisti – Topology

Reference

Cisco SMM | TOPOLOGY

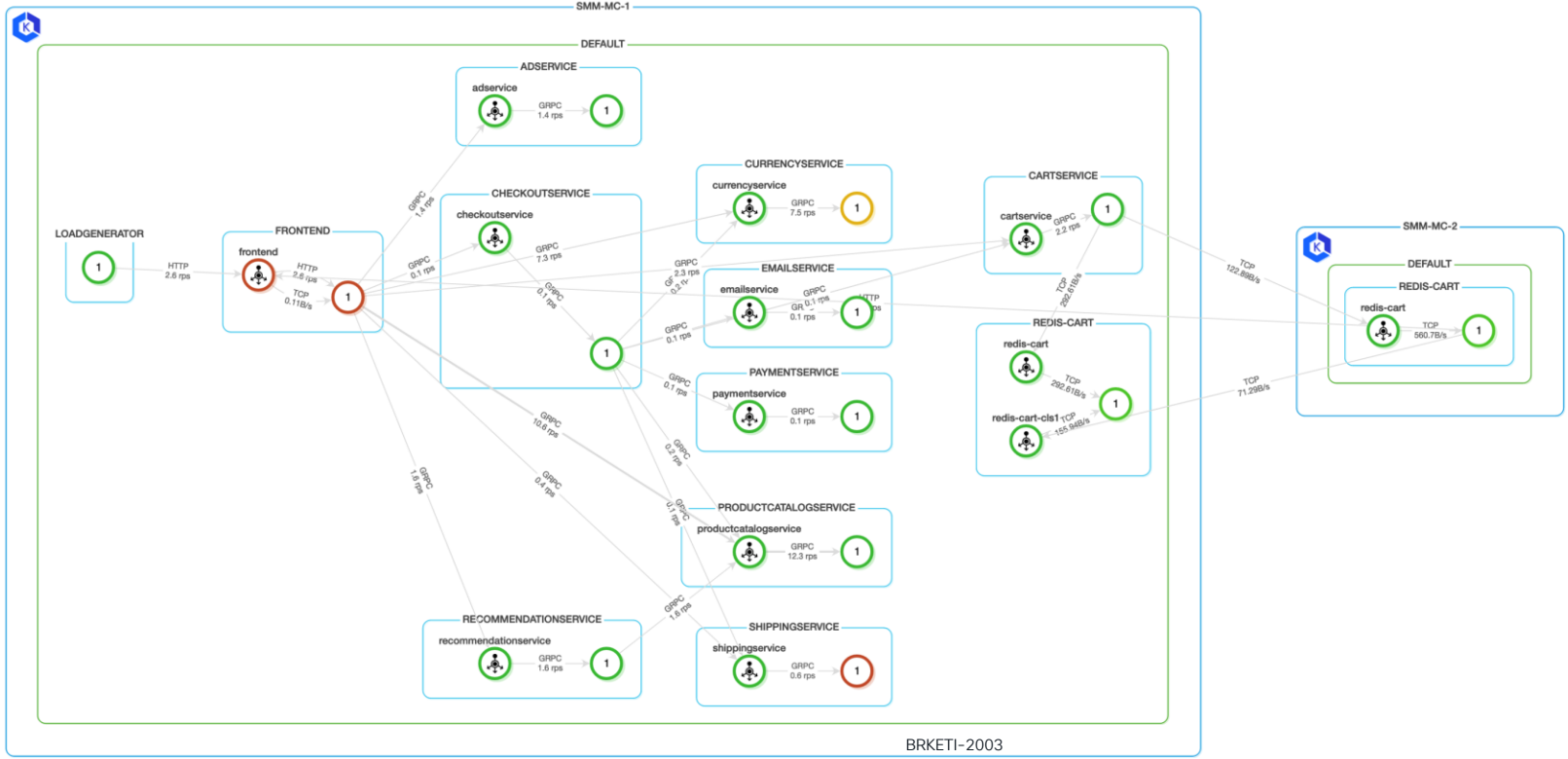
NAMESPACES (1): default

RESOURCES: workloads services apps namespaces clusters

EDGE LABELS: request rate / throughput

TIMELINE

HTTP



Summary

- Check out Cisco Calisti – Get started for free (Up to 10 nodes and 2 clusters): https://calisti.app/
- There are many options for connecting workloads in multiple Kubernetes clusters – we just touched on a few
  - Network Service Mesh - https://networkservicemesh.io/
- Many users leverage multicluster connectivity for cross-cluster load-balancing of services
- For specialized per-service cross-cluster connectivity, special care must be taken to select a solution that provides a balance of use-case flexibility and operational supportability

Technical Session Surveys

- Attendees who fill out a minimum of four session surveys and the overall event survey will get Cisco Live branded socks!
- Attendees will also earn 100 points in the Cisco Live Game for every survey completed.
- These points help you get on the leaderboard and increase your chances of winning daily and grand prizes.

Cisco Learning and Certifications

From technology training and team development to Cisco certifications and learning plans, let us help you empower your business and career. www.cisco.com/go/certs

Pay for Learning with Cisco Learning Credits

(CLCs) are prepaid training vouchers redeemed directly with Cisco.

Learn

Cisco U.
IT learning hub that guides teams and learners toward their goals

Cisco Digital Learning
Subscription-based product, technology, and certification training

Cisco Modeling Labs
Network simulation platform for design, testing, and troubleshooting

Cisco Learning Network
Resource community portal for certifications and learning

Train

Cisco Training Bootcamps
Intensive team & individual automation and technology training programs

Cisco Learning Partner Program
Authorized training partners supporting Cisco technology and career certifications

Cisco Instructor-led and Virtual Instructor-led training
Accelerated curriculum of product, technology, and certification courses

Certify

Cisco Certifications and Specialist Certifications
Award-winning certification program empowers students and IT Professionals to advance their technical careers

Cisco Guided Study Groups
180-day certification prep program with learning and support

Cisco Continuing Education Program
Recertification training options for Cisco certified individuals

Here at the event? Visit us at **The Learning and Certifications lounge at the World of Solutions**

Continue your education

- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer meeting
- 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

Thank you

cisco Live!

ALL IN

#CiscoLive