Multi-Cluster Communication with Linkerd

Max Körbächer | Liquid Reply

Let’s connect!

maxkoerbaecher mkoerbi mkorbi

Co-Founder & Manager Cloud Native Engineering

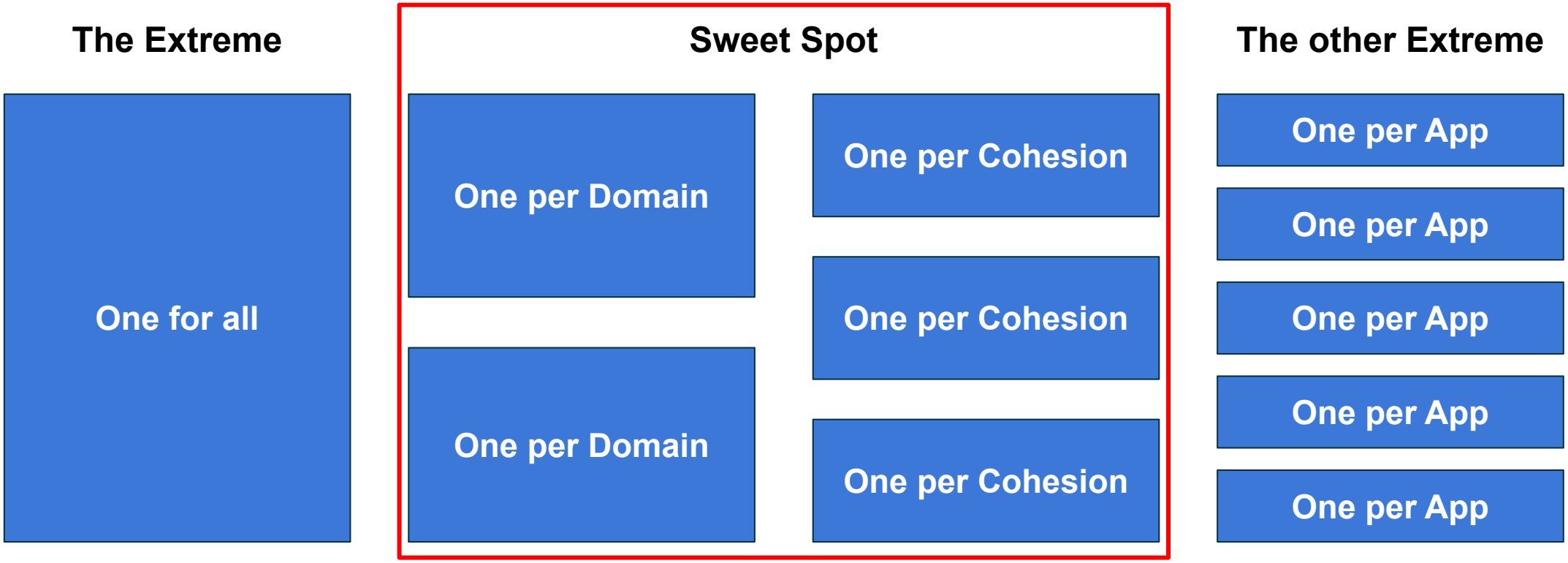
@ Liquid Reply

Kubernetes SIG Release since v1.17

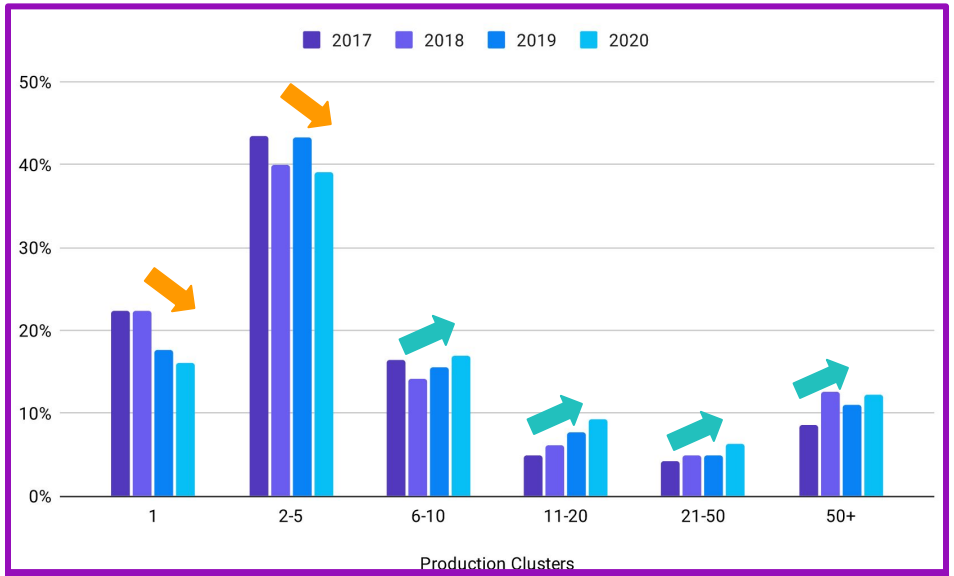
If not bringing K8s to customer I’m playing with drones and crafting the nativecloud.dev newsletter

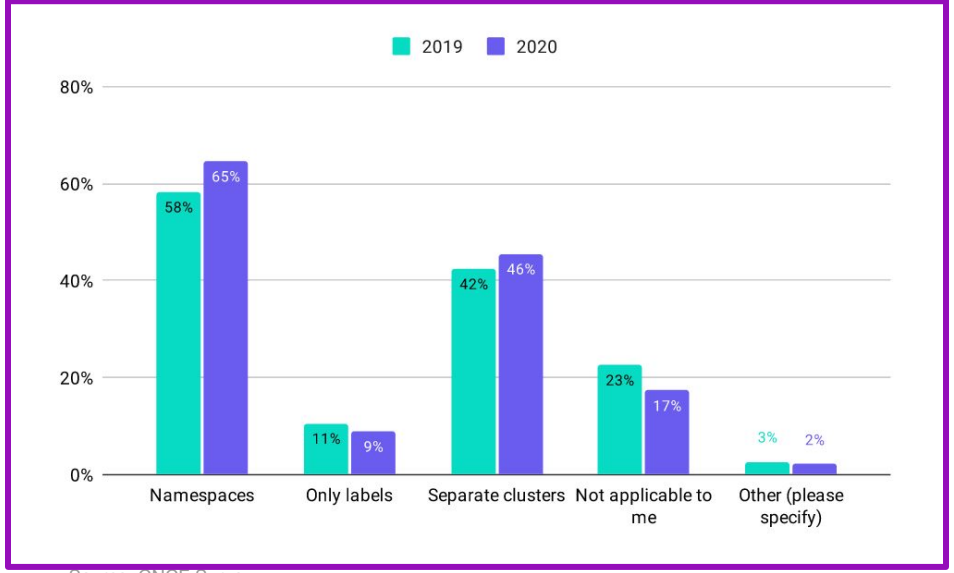

Multi-Cluster
Communication with Linkerd
One of Many Approaches

Kubernetes Clusters comes with many shapes and sizes



Adopting clusters in prod ...





Source: CNCF Survey

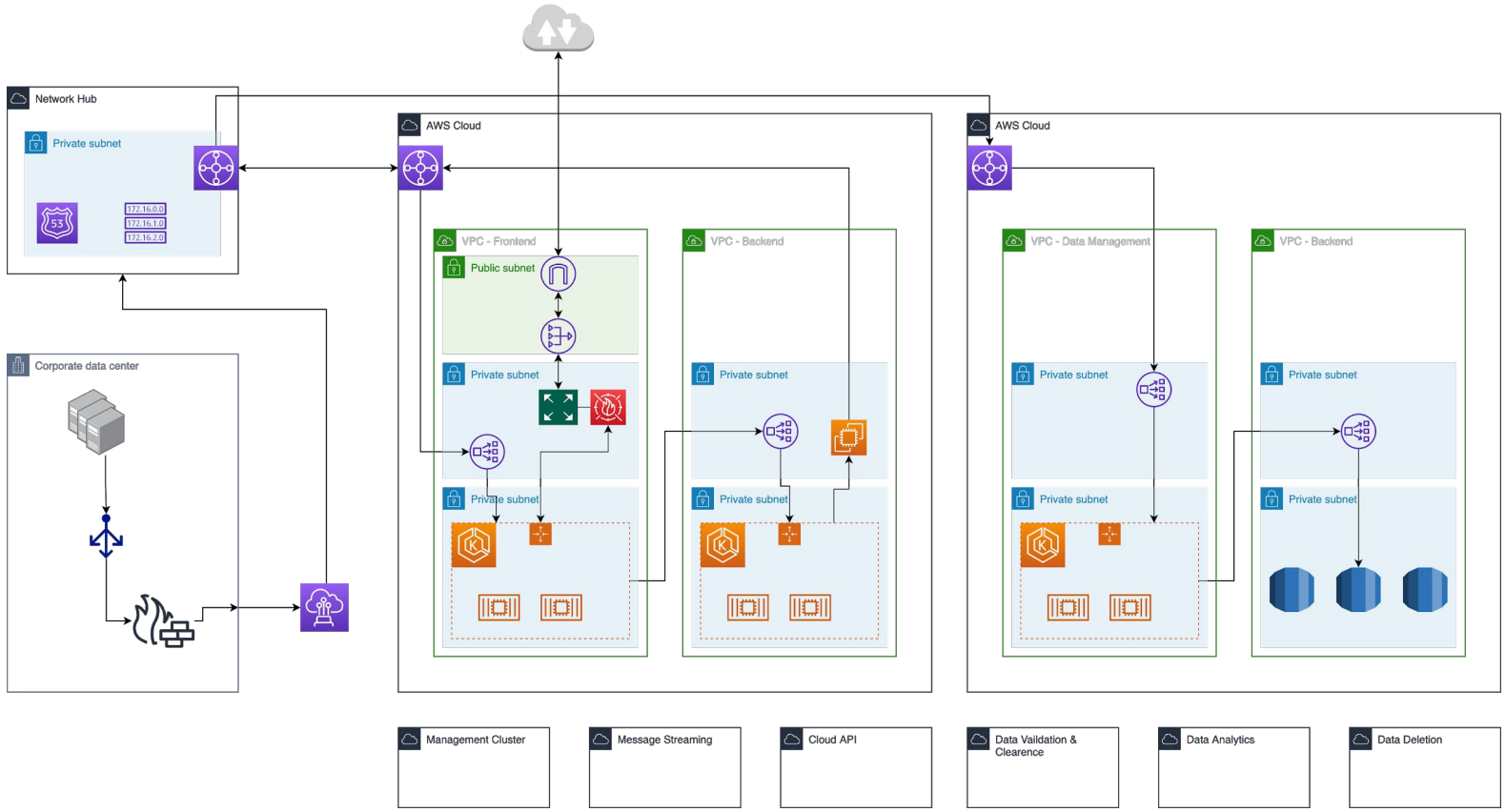
The adoption of K8s clusters in productive scenarios are growing, at the same time the average amount of clusters in production grows too.

Furthermore, multiple teams working together think more about isolation ether through namespaces or cluster separation.

But they struggle with topics like: complexity, security, cultural change, monitoring, networking, logging, lack of training and many more…

This is where our, and as it looks like, the journey of many others starts.

In many scenarios we see the same sprawl



1. An unnecessary amount of clusters
    - communication between system components goes “out of the cluster”
    - can’t call a service by it’s service name
2. Each service has unique restrictions
3. Certificate management and network encryption without fun
4. Changes on the system often requires changes in the infrastructure
5. To many failure/break points - impossible debugging

It is a stable cloud architecture, but it is not a cloud native architecture.

Introducing Linkerd Multi-Cluster

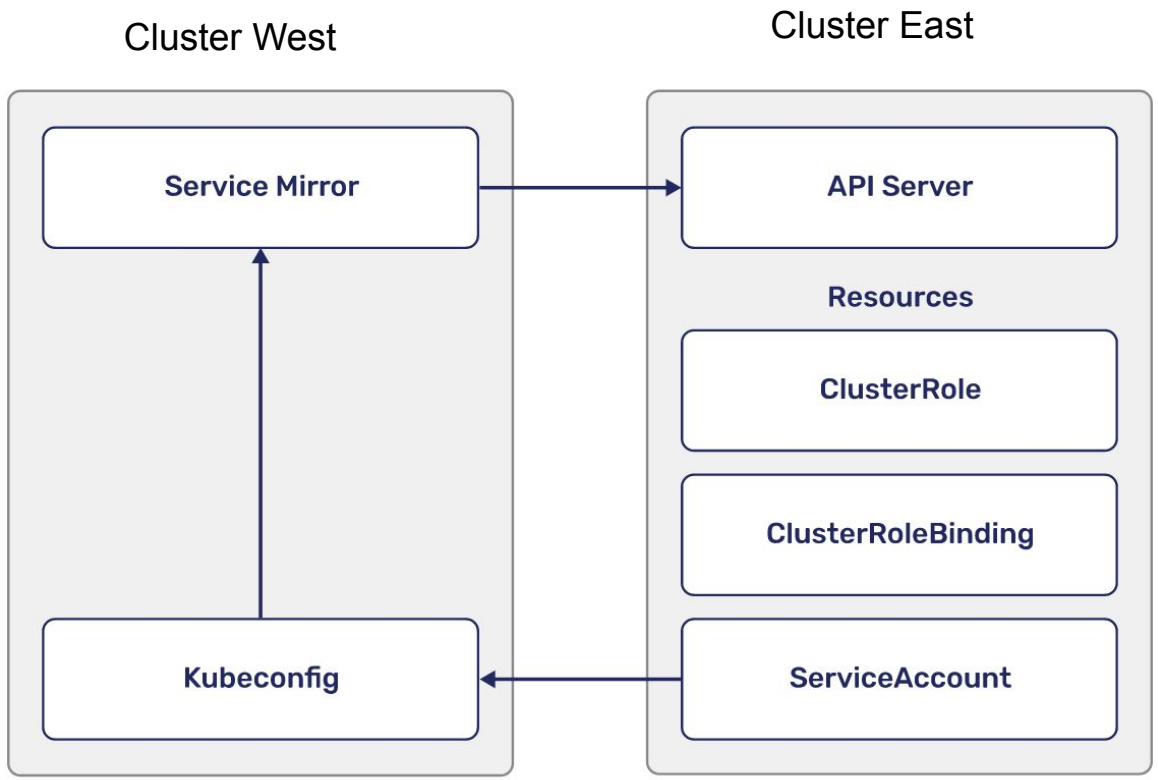
- Linkerd doesn't do multi-cluster out of the box
- Multicluster is a useful extensions and lives aside other useful extensions like viz for metrics and jaeger for tracing
- Install multicluster via the linkerd CLI
- Important is the trust anchor, a certificate, which is shared between the go to be linked clusters

```
NAMESPACE             NAME                                          READY
default               yelb-ui-6bcc9b9d84-zjv68                      2/2
kube-system           coredns-74ff55c5b-5tk4d                       1/1
kube-system           coredns-74ff55c5b-xkm6q                       1/1
kube-system           etcd-kind-control-plane                       1/1
kube-system           kindnet-f8wmh                                 1/1
kube-system           kube-apiserver-kind-control-plane             1/1
kube-system           kube-controller-manager-kind-control-plane    1/1
kube-system           kube-proxy-vffgf                              1/1
kube-system           kube-scheduler-kind-control-plane             1/1
linkerd-multicluster  linkerd-gateway-6f94dc94bb-rdjtq              2/2
linkerd-multicluster  linkerd-service-mirror-west-6556c8597b-zrk2h  2/2
linkerd-viz           grafana-76ccc5f488-t55tf                      2/2
linkerd-viz           metrics-api-587695476f-vwkf2                  2/2
linkerd-viz           prometheus-d688b44c-kdthr                     2/2
linkerd-viz           tap-6458f6949b-bk42t                          2/2
linkerd-viz           tap-injector-6d7cc57598-hzspq                 2/2
linkerd-viz           web-879bf6956-mz79l                           2/2
linkerd               linkerd-controller-7fd95568f9-jpgp2           2/2
linkerd               linkerd-destination-85f9f56b47-9qg9s          2/2
linkerd               linkerd-identity-8f6cb85ff-wl6q6              2/2
linkerd               linkerd-proxy-injector-6c544986f4-62vfg       2/2
linkerd               linkerd-sp-validator-b7b867cf4-srcff          2/2
local-path-storage    local-path-provisioner-78776bfc44-2gp4p       1/1
```

How to Linkerd Multi-Cluster

Step 2: Link the clusters

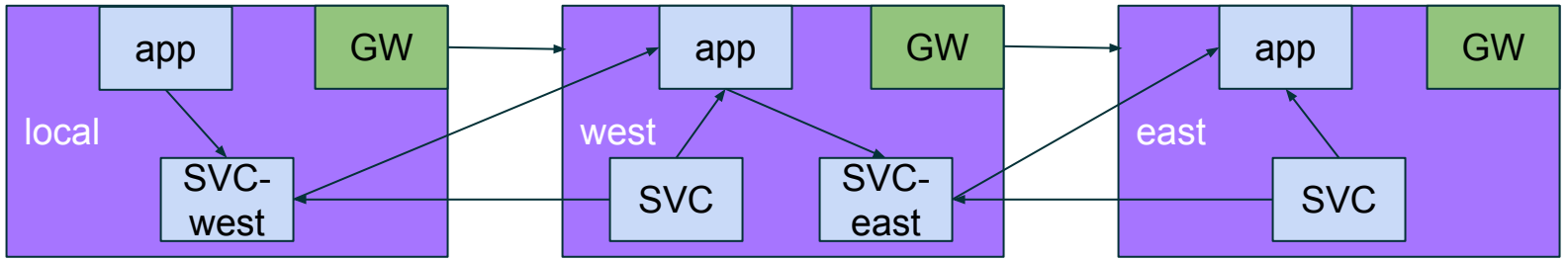
- linkerd wraps up a role, service account and relevant connection data from east cluster and inject this into the west cluster as kubeconfig
- This allows to propagate via a Service Mirror the service to the east cluster



Link clusters and export SVC

Step 2: Link the clusters

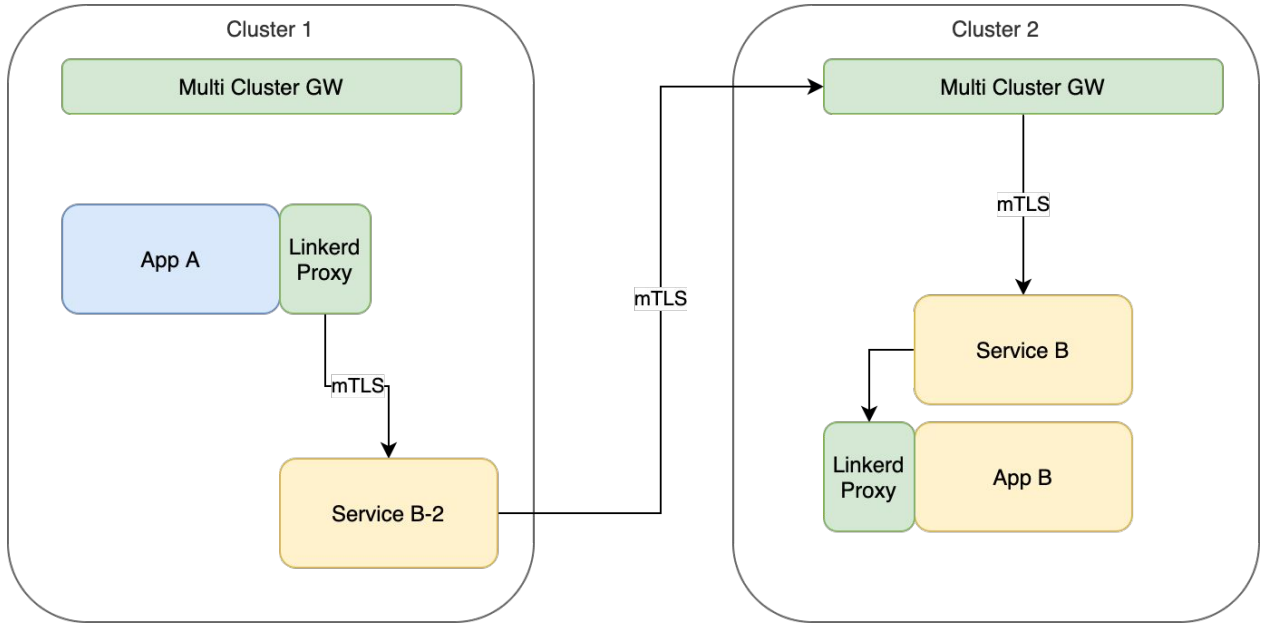
```
linkerd --context=east multicluster install | kubectl --context=east apply -f -
linkerd --context=west multicluster install | kubectl --context=west apply -f -


linkerd --context=east multicluster link --cluster-name east |
 kubectl --context=west apply -f -

# annotate your service with mirror.linkerd.io/exported=true
# this exports your service to the linked cluster
```



How to Linkerd Multi-Cluster

Step 3: Export the service

- Now that the clusters are aware of each other the Multi Cluster Gateway is ready to receive the requests
- To make a service being mirrored the service needs to be annotated:
  - `mirror.linkerd.io/exported=true`
- This duplicates the service into the other cluster with a “-clustername”



Demo

For the demo I created a similar setup as seen before

- Local: kind cluster - frontend
- West: eks cluster - eu-west-1 - backend
- East: eks cluster - eu-west-3 - data management/DB

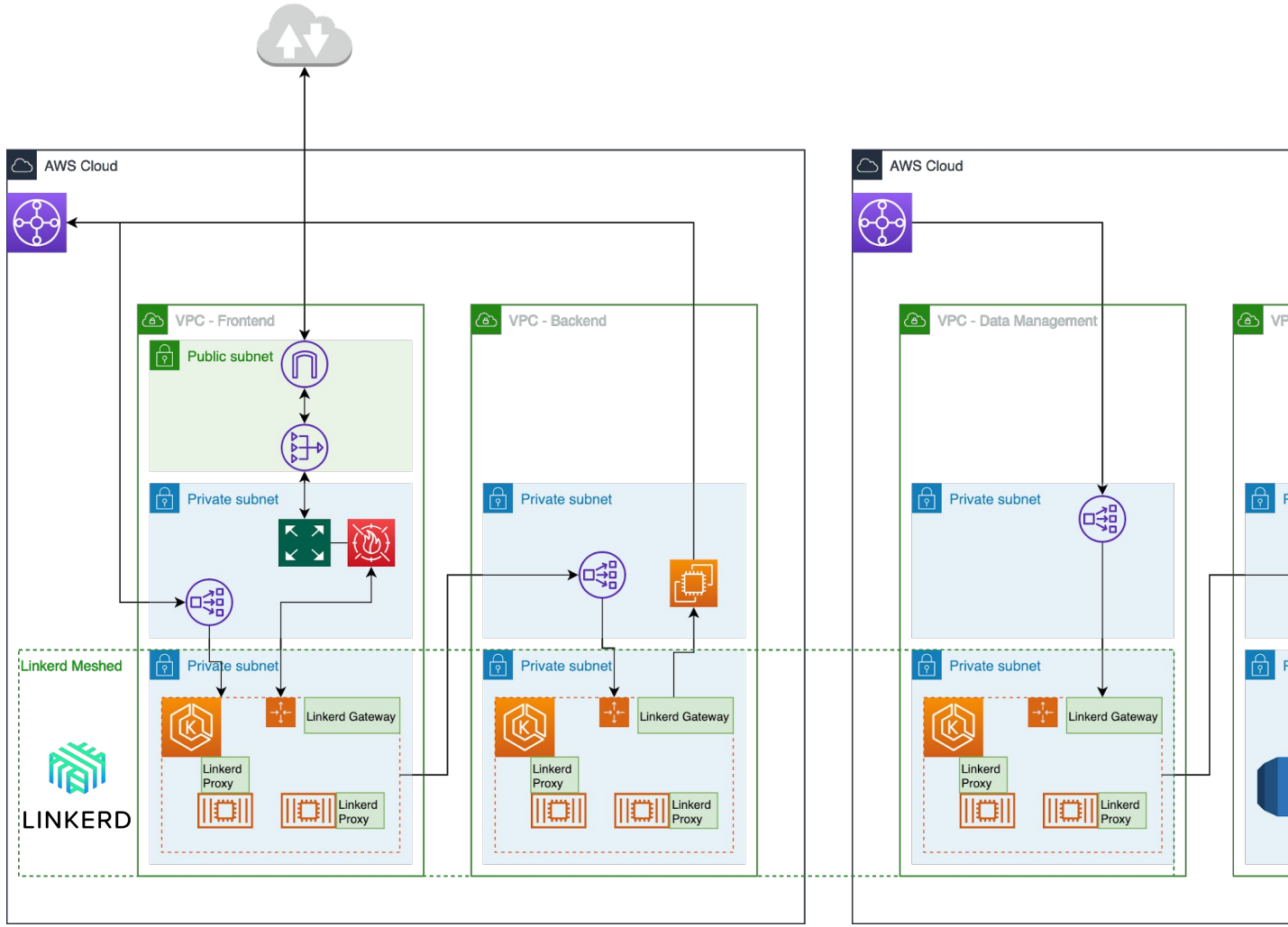
Chain linked the Clusters

Our clusters got linked just in one direction
Frontend -> Backend -> Data Management

In another way around the services were exported from
Data Management -> Backend -> Frontend

Don’t miss:

- to install multicluster with --api-server-address otherwise linkerd will not find the api server in a heavily isolated network
- to annotate the to be exported services `mirror.linkerd.io/exported=true`



Positive Effect: Insights out of the box

Deployments

Deployment	Meshed	Success Rate	RPS	P50 Latency	P95 Latency	P99 Latency	Grafana
redis-server	1/1	100.00%	0.3	1 ms	1 ms	1 ms	
yelb-appserver	1/1	100.00%	0.3	1 ms	1 ms	1 ms	

Pods

Pod	Meshed	Success Rate	RPS	P50 Latency	P95 Latency	P99 Latency	Grafana
redis-server-7bc9f68549-xh8wp	1/1	100.00%	0.3	1 ms	1 ms	1 ms	
yelb-appserver-856586c95-j8d5d	1/1	100.00%	0.3	1 ms	1 ms	1 ms	

TCP

Pod	Meshed	Connections	Read Bytes / sec	Write Bytes / sec	Grafana
redis-server-7bc9f68549-xh8wp	1/1	1	32B/s	290.52B/s	
yelb-appserver-856586c95-j8d5d	1/1	1	31.67B/s	433.88B/s	

Success Rate: percentage of successful requests during a time window

RPS: how much demand is placed on the service/route

Latency: times taken to service requests per service/route

Connections: current number of connections

Read/Write Bytes/sec: avrg written/read bytes per second

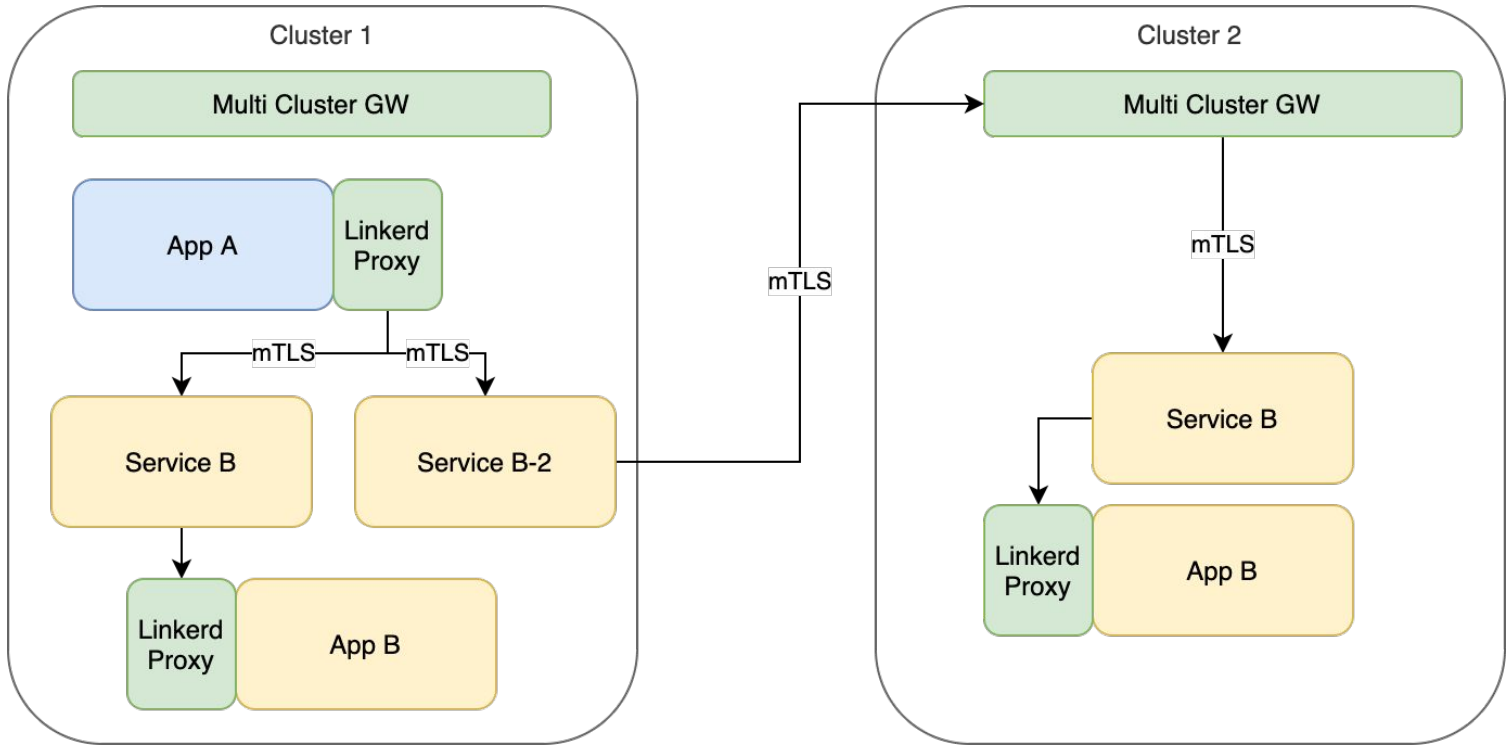
When Multi-Cluster makes Sense?

1. **Full fallback/blue-green cluster**
   When you have another full blown K8s cluster as fall back or as blue/green cluster
2. **A hand full of clusters who needs to talk**
   The amount of clusters shouldn't be to much, however it obviously depends on the amount of apps too
3. **Hybrid environment extension**
   Extending your on prem service to an cloud provider or vise versa e.g. to process sensitive data just where they are allowed to exist

Traffic Split

A Fallback Solution

```
apiVersion:
split.smi-spec.io/v1alpha1
kind: TrafficSplit
metadata:
 name: service-b
 namespace: default
spec:
 service: service-b
 backends:
 - service: service-b
   weight: 50
 - service: service-b-2
   weight: 50
```



What Multi-Cluster gives you

Work once on the network

You have to get your network once right, afterwards you can manage it by yourself

Security

Everything communicates via mTLS

Kubernetes First

What happens in K8s stays in K8s. In other words: your services doesn't need to know if they communicate with something remote in another cluster or else where

Declaratively Transparent

What is defined counts, no external (eg. infra team) dependencies, easy to understand the given configuration.

(in some cases) No Single Point of Failure

No cluster (upgrade) is perfect, so don't rely on it.

Lessons learned

1. Existing apps need to adjust their calling service endpoint from e.g. kubecon-svc to kubecon-svc-theOtherCluster

2. Linkerd upgrades can be challenge - just have a look at the last two versions which have an own upgrade guide

3. Sometimes SVC didn't got updated after rollout of a new version - however, we couldn't reproduce this

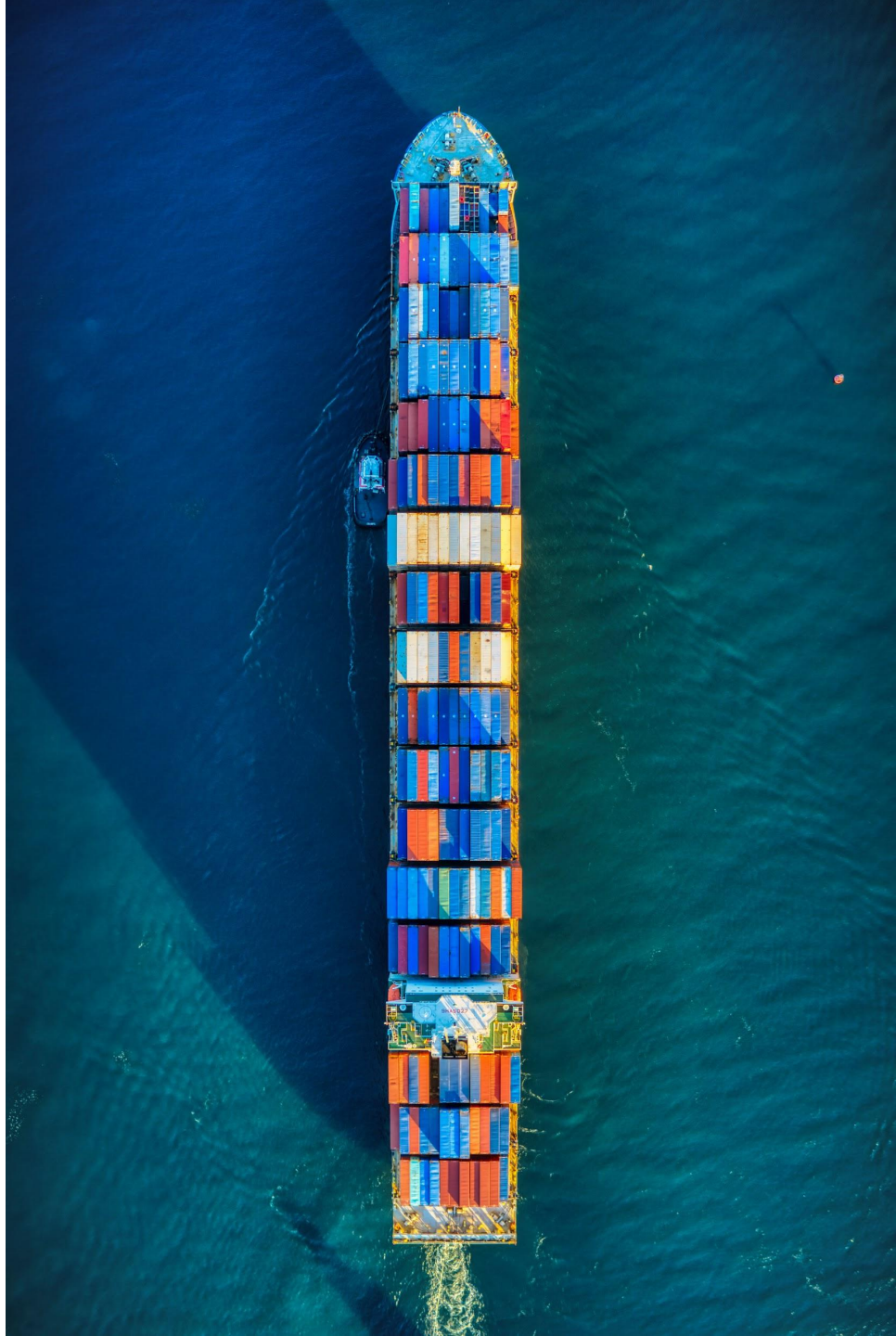
4. Service Meshes are a rabbit hole, even when they are easy to be deployed - think twice

5. Don't let your service names get to long…
   - a-longer-service-name-eu-west-1-domain-department-squad-cost-center-prod is still ok ...

Wrap Up

Some things to definitely check out:

1. Set your Linkerd installation to *--ha*
2. Custom *--registry* flag is a life saver
3. *--admin-port* a plus for selling linkerd to security
4. *--proxy-cpu/memory-request/limit* for the cluster hygiene
5. *linkerd.io/inject: enabled* per namespace vs per pod
   a. in pre-production and production we auto-inject per namespace
   b. in dev & test you have to add in at the deployment
6. linkerd needs *NET_RAW & NET_ADMIN* capabilities - when securing clusters, that's a topic



DevOpsCon

THE ONLINE CONFERENCE