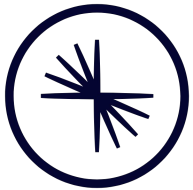
Improving the reliability of Kubernetes load balancers

Alexander Constantinescu



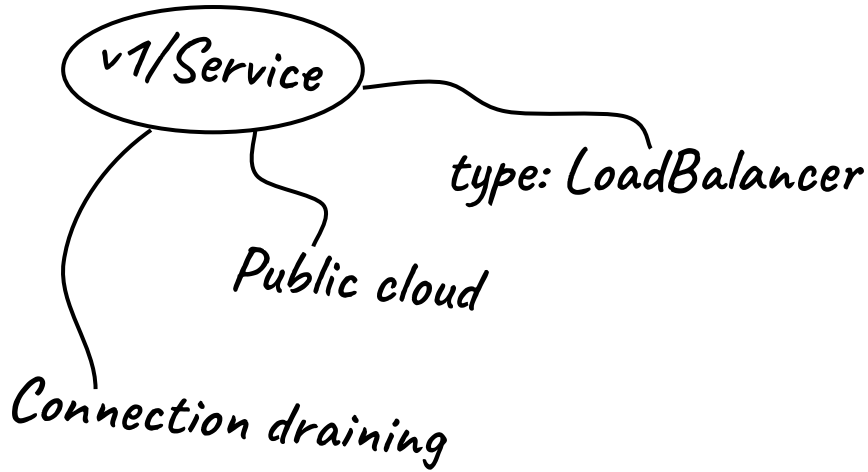
- Background
- Problem
- Solution
- Future work

Background

"Kubernetes load balancers"? Let's be specific...

~~MetalLB~~

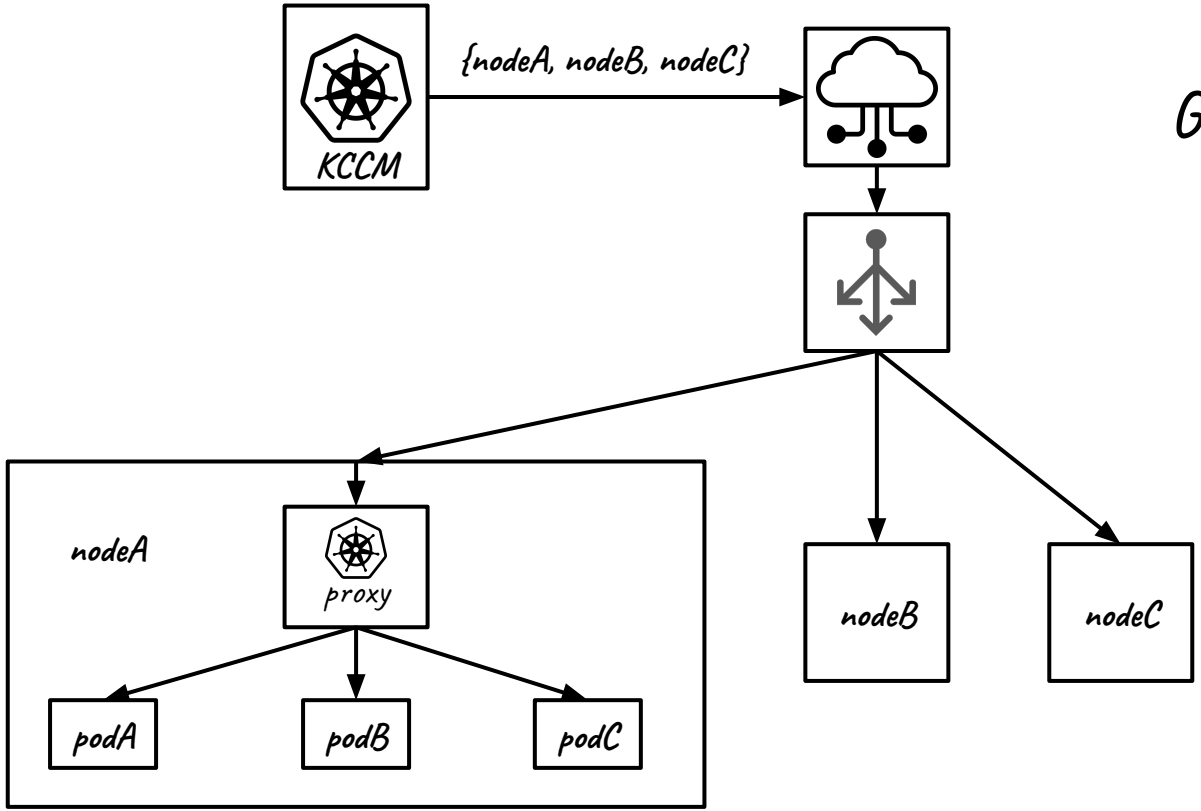
~~ingress.*~~



Background

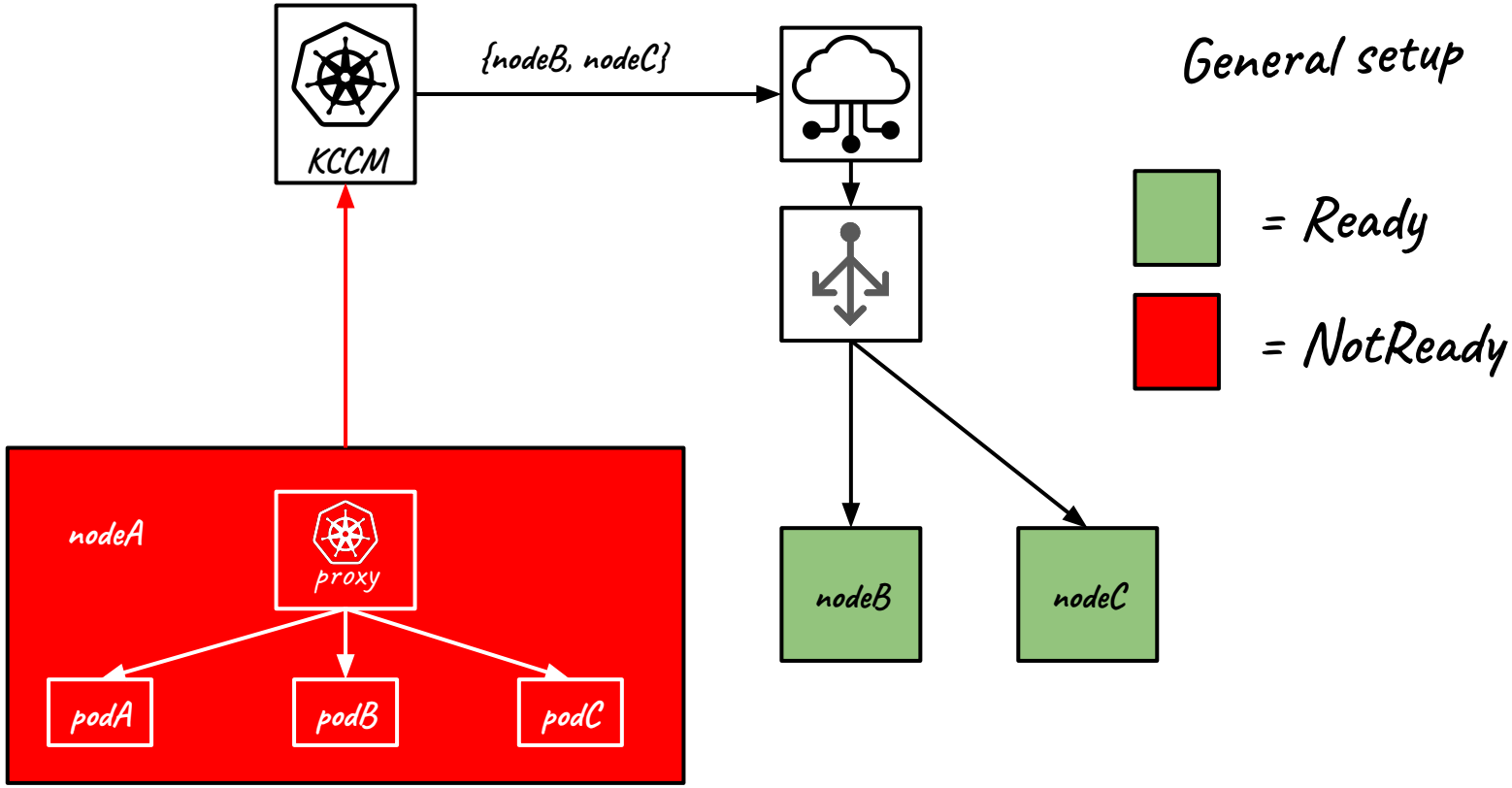
How do “v1/service load balancers” work today?

Background - configuring a service load balancer

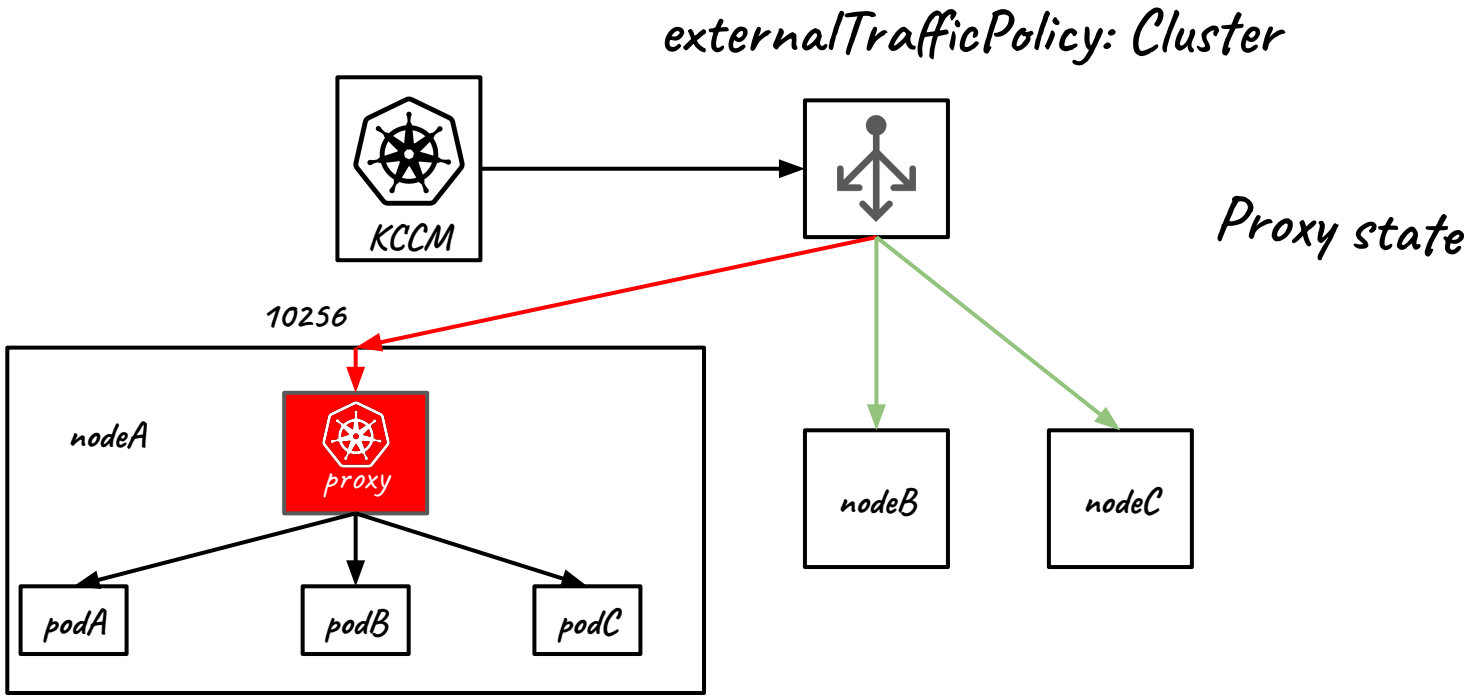


General setup

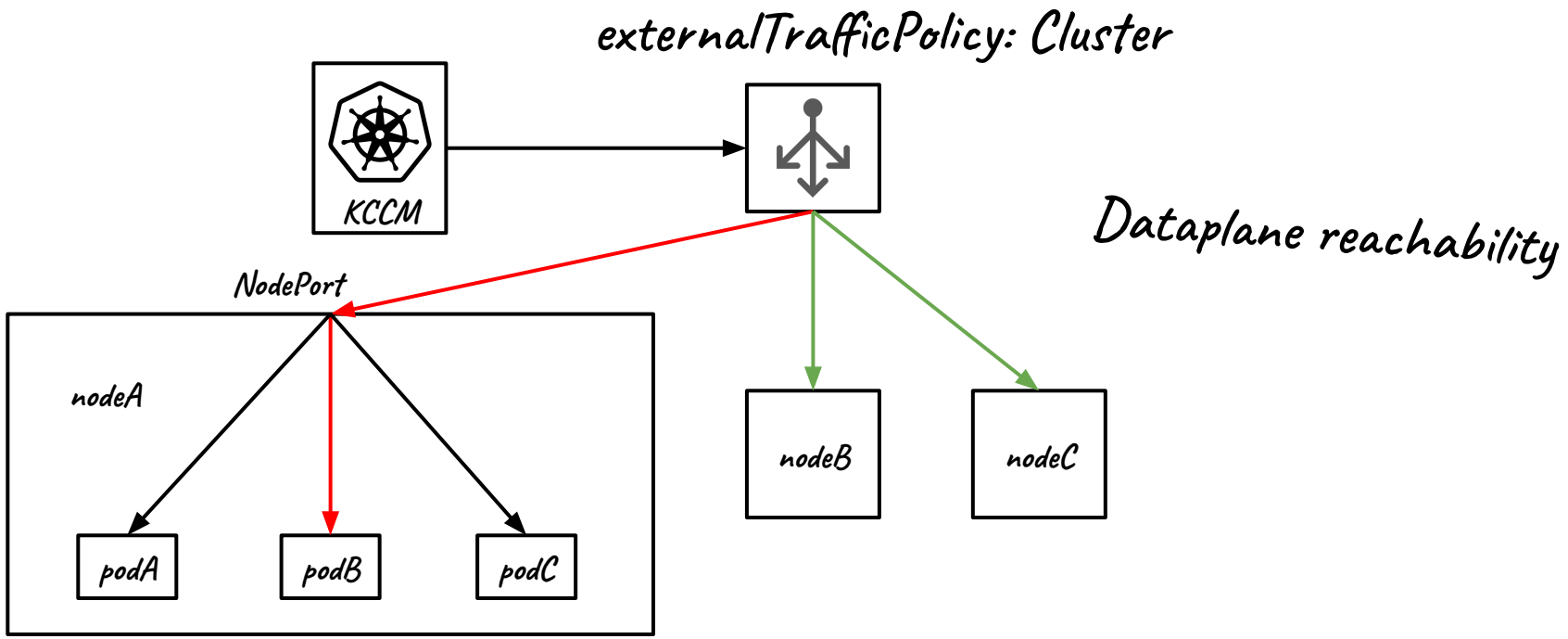
Background - configuring a service load balancer



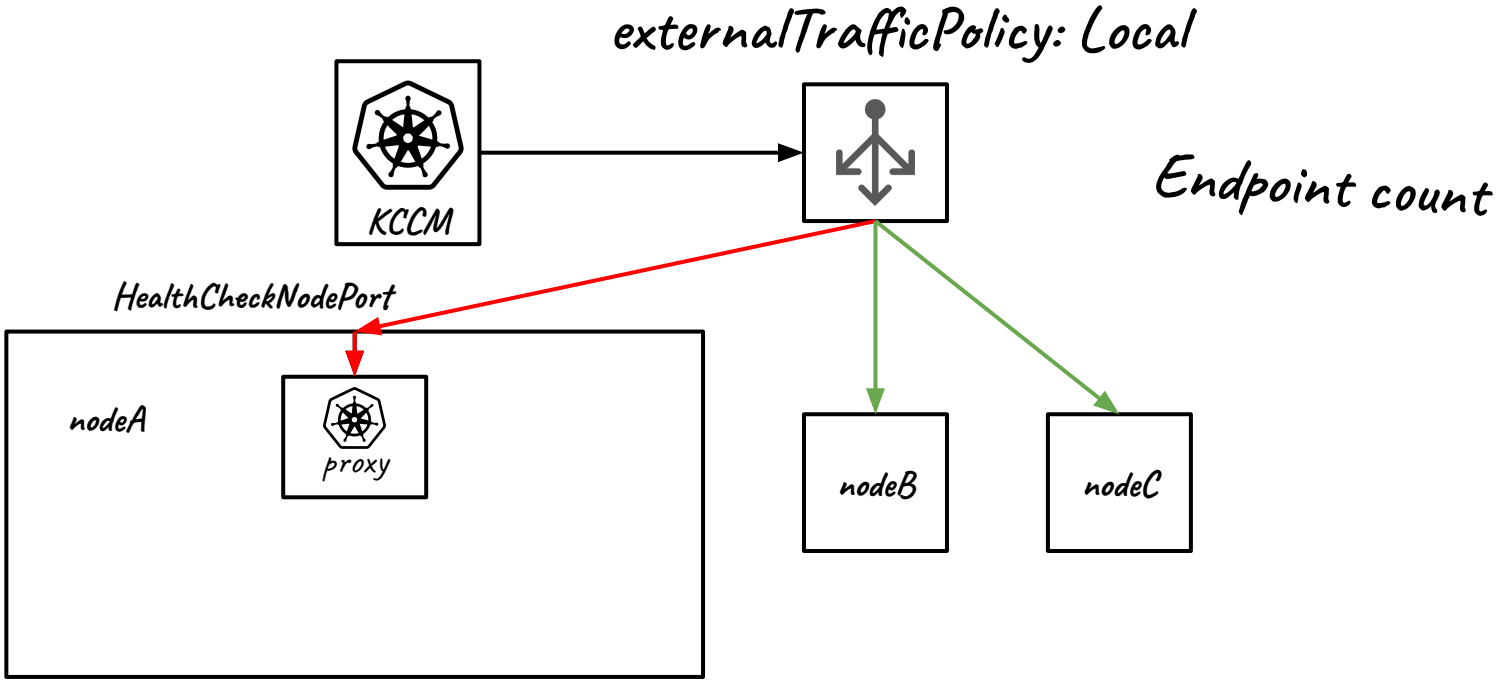
Background - health checking a service load balancer



Background - health checking a service load balancer



Background - health checking a service load balancer



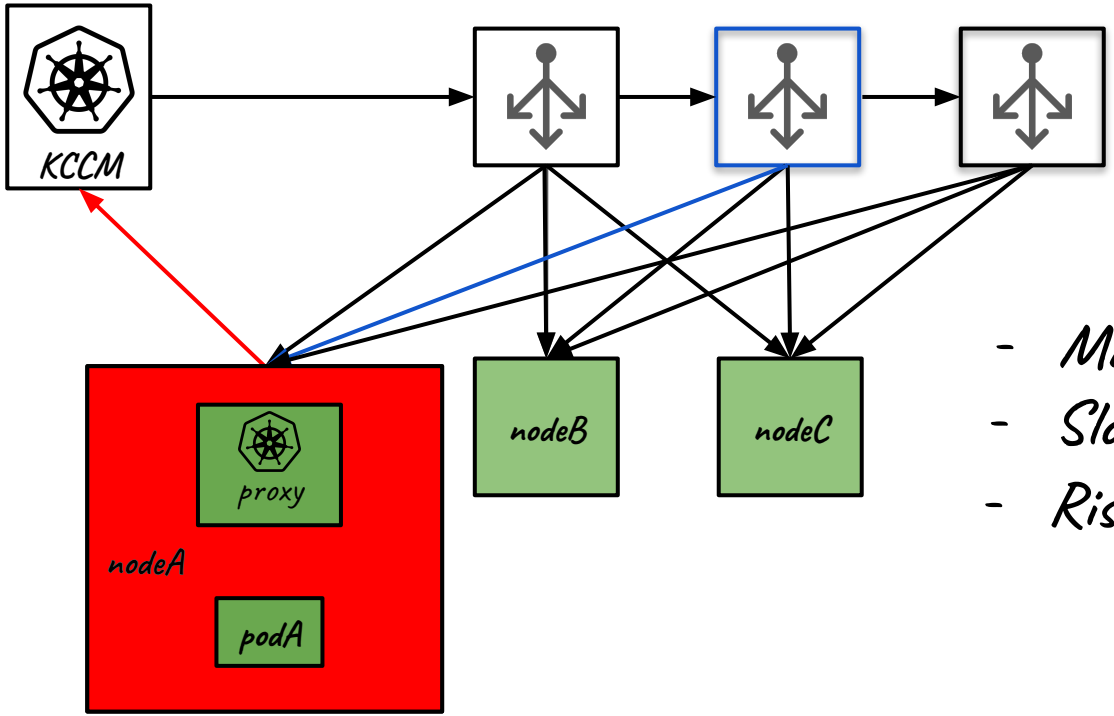
Background - connection draining

= "block new connection, allow existing to terminate"

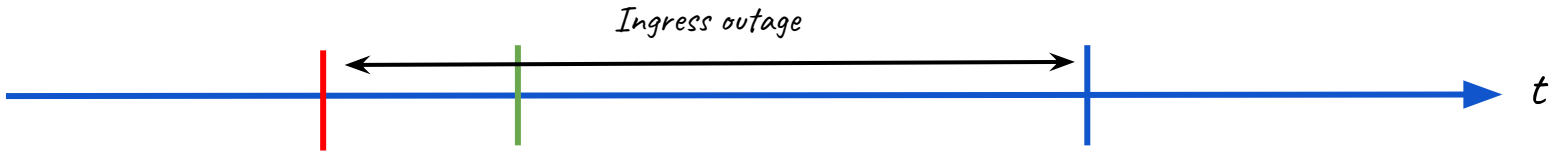
Problem

So, what are the problems?

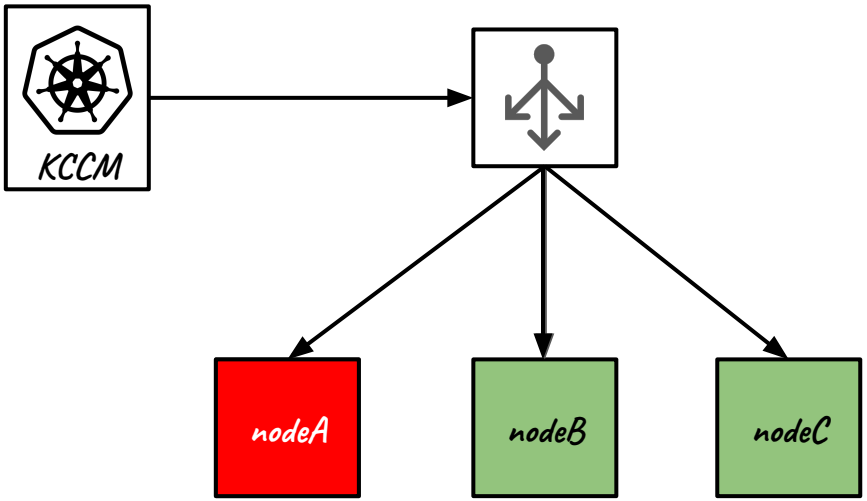
Problem - impact on performance & ingress



- Many cloud API calls
- Slow to sync
- Risk de-syncing Kube vs. cloud state



Problem - impact on connectivity



All connections

Solution

What do we want instead?

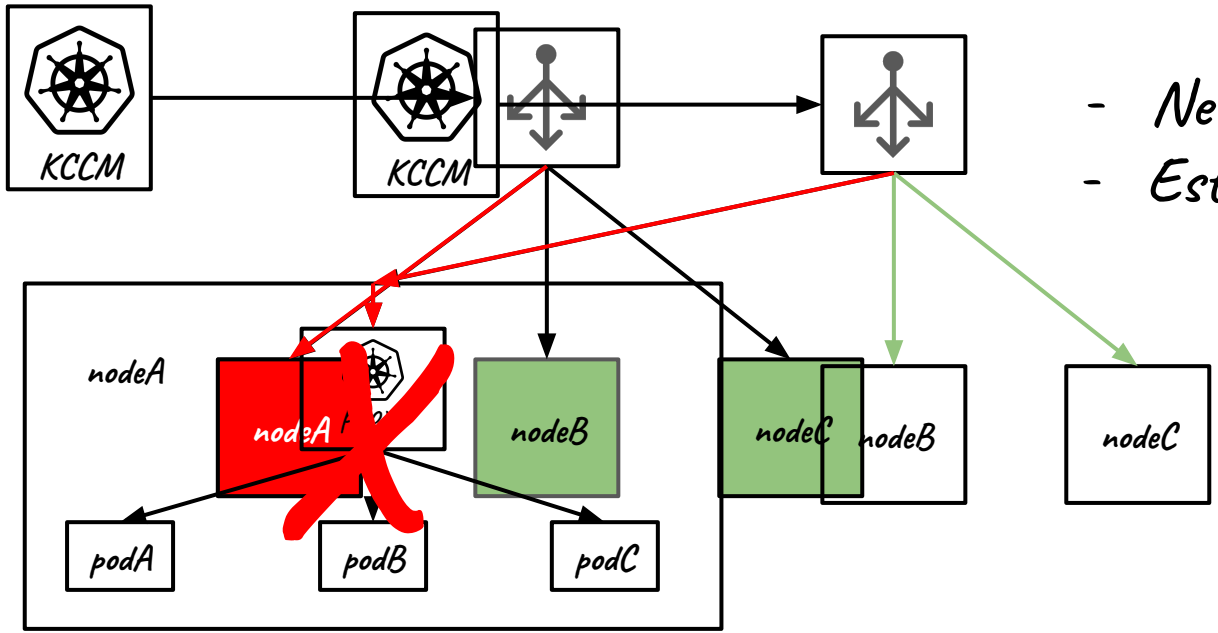
Solution

Kubernetes Nodes & load balancers...

- Readiness
- Terminating

Solution

- Ready => no impact on load balancer config
- Terminating => connection drain



- New connections
- Established

Solution

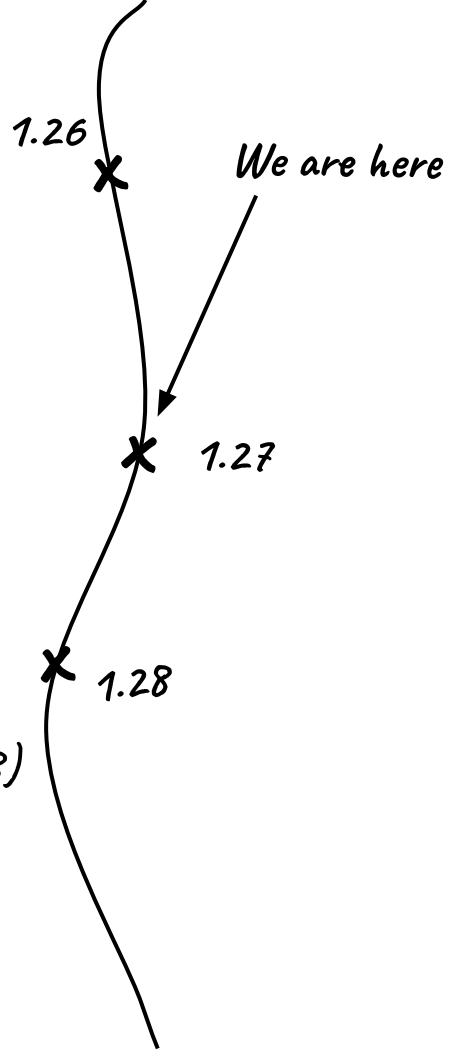
Where are we?

Solution

[KCCM] eTP:Local - stop re-syncing LBs on node readiness (1.26)

KEP-3458: [KCCM] to stop syncing LBs on node readiness (beta 1.27)

KEP-3836: [Kube-proxy] Connection drain terminating nodes (alpha 1.28)

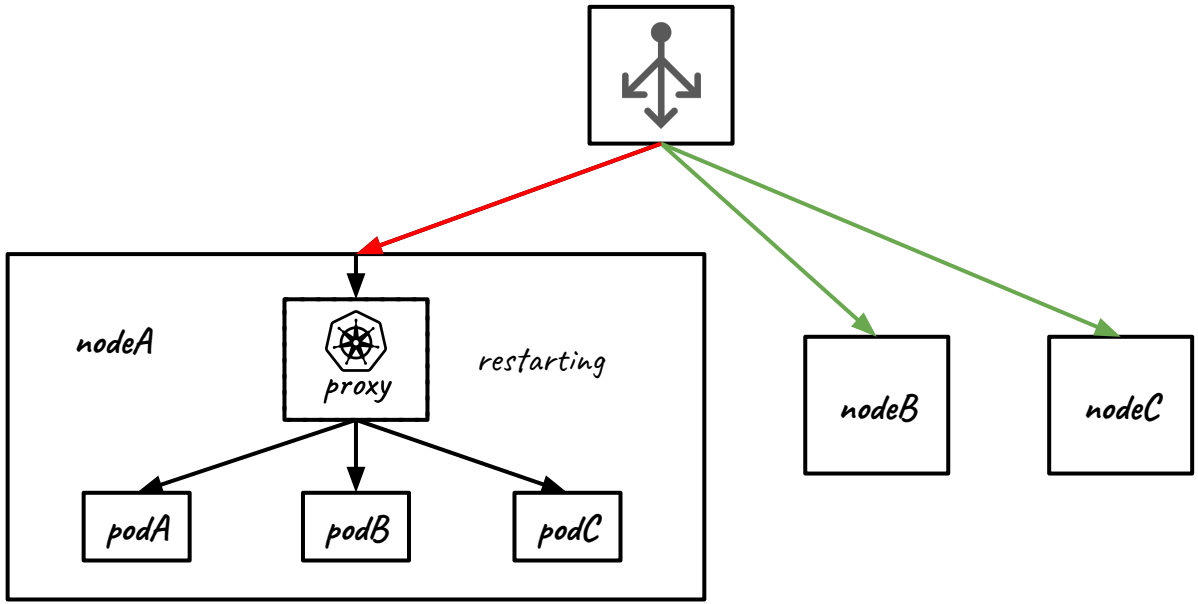


Future work

What can we still improve?

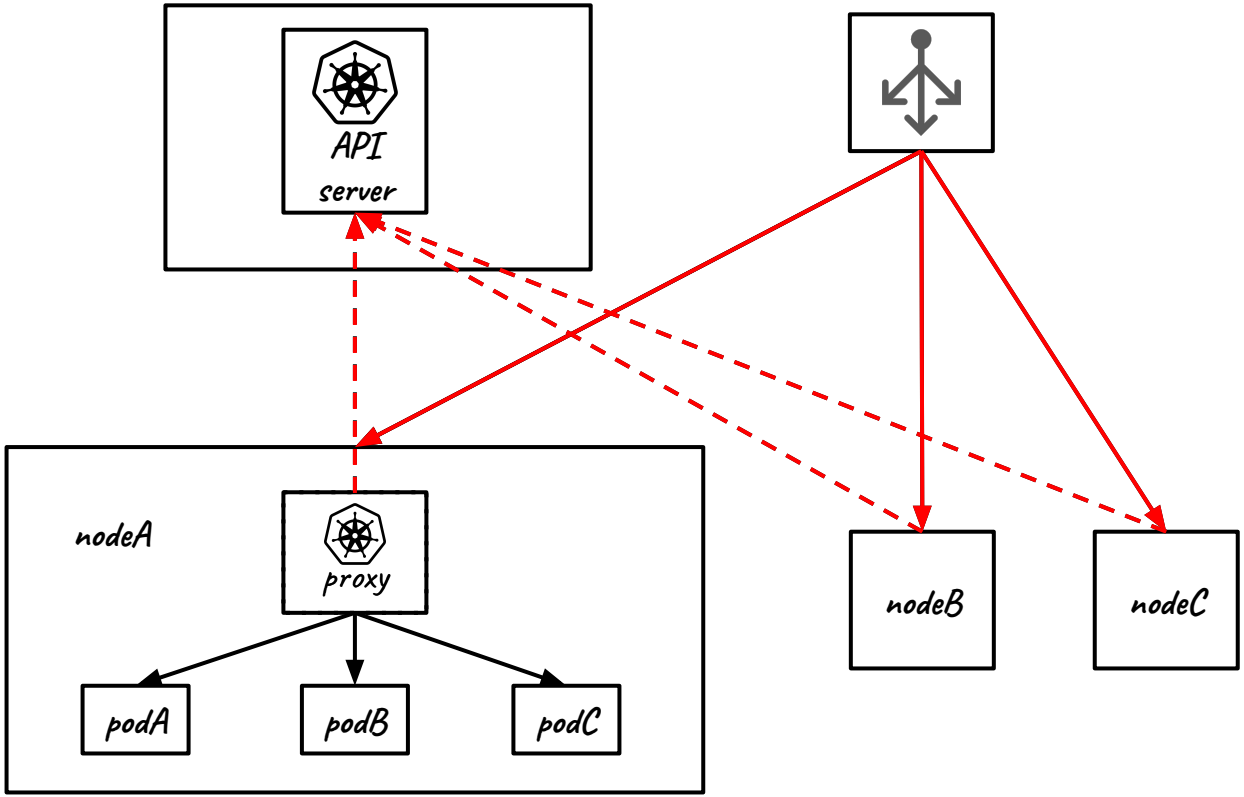
Future work

De-couple health check server from service-proxy (eTP:Local)



Future work

Kube-proxy health "majority report"



Future work

Recommendations?

Recommendations

- Cloud provider? Think about your LB health checks
  - Profit from KEP-3836
- Service-proxy? Think about enabling LBs to connection drain
  - Cross service-proxy usability alignment

Thank you!

...questions?