Kubernetes the Very Hard Way

Laurent Bernaille

Staff Engineer, Infrastructure

@lbernail

Datadog

Over 350 integrations
Over 1,200 employees
Over 8,000 customers
Runs on millions of hosts
Trillions of data points per day

10000s hosts in our infra
10s of k8s clusters with 50-2500 nodes
Multi-cloud
Very fast growth

Why Kubernetes?

Dogfooding

Improve k8s integrations

Immutable

Move from Chef

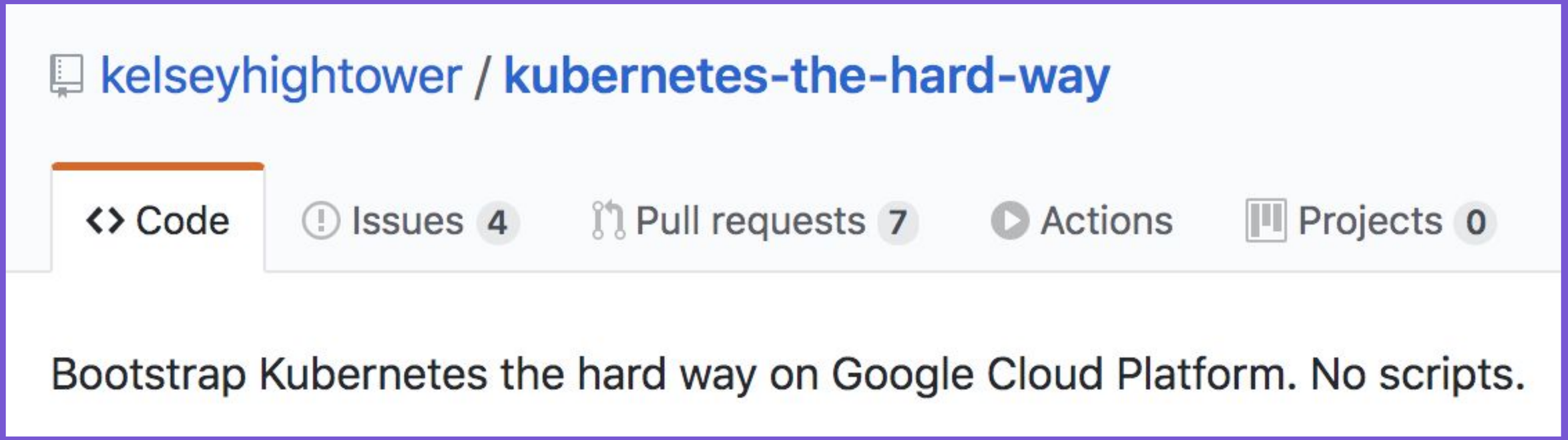
Multi Cloud

Common API

Community

Large and Dynamic

The very hard way?



It was *much* harder

This talk is about the fine print

“Of course, you will need a HA master setup”

“Oh, and yes, you will have to manage your certificates”

“By the way, networking is slightly more complicated, look into CNI / ingress controllers”

What happens after “Kube 101”

1. Resilient and Scalable Control Plane
2. Securing the Control Plane
    a. Kubernetes and Certificates
    b. Exceptions?
    c. Impact of Certificate Rotation
3. Efficient networking
    a. Giving pod IPs and routing them
    b. Ingresses: Getting data in the cluster

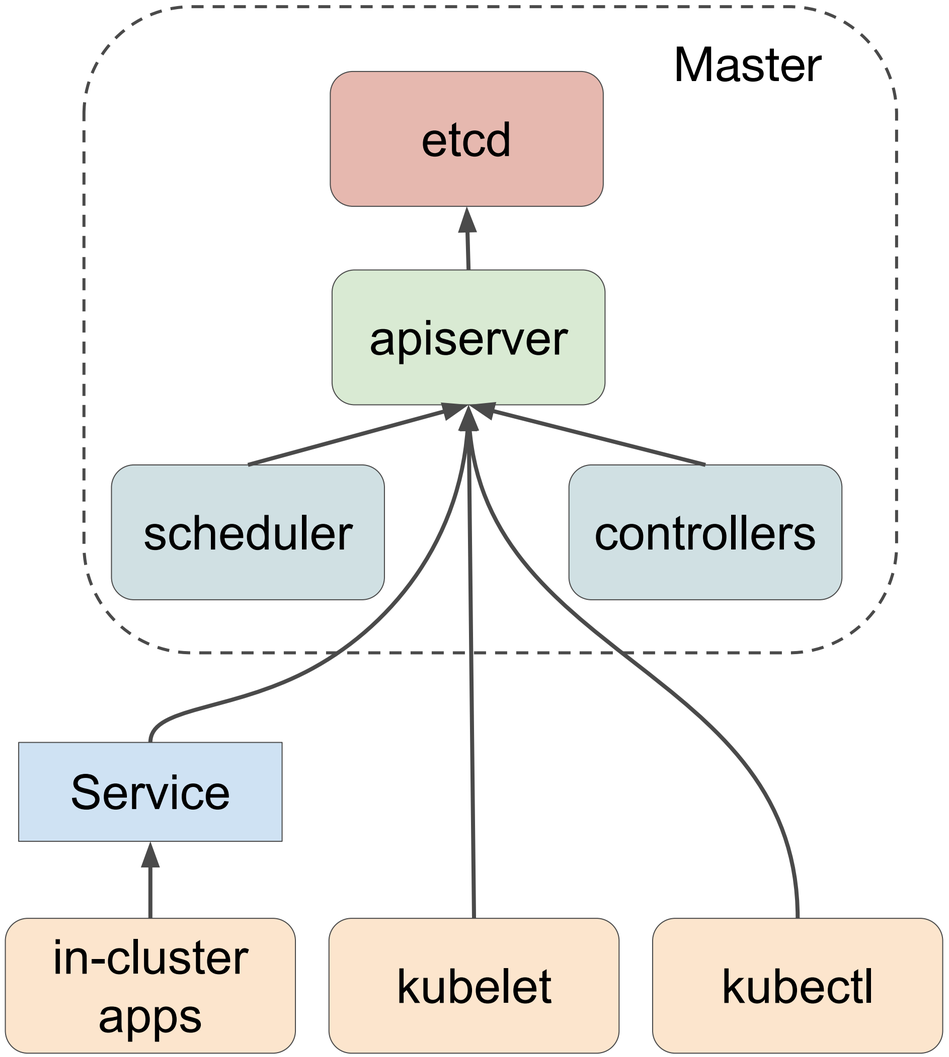
What happens after “Kube 101”

1. Resilient and Scalable Control Plane
2. Securing the Control Plane
    a. Kubernetes and Certificates
    b. Exceptions?
    c. Impact of Certificate Rotation
3. Efficient networking
    a. Giving pod IPs and routing them
    b. Ingresses: Getting data in the cluster

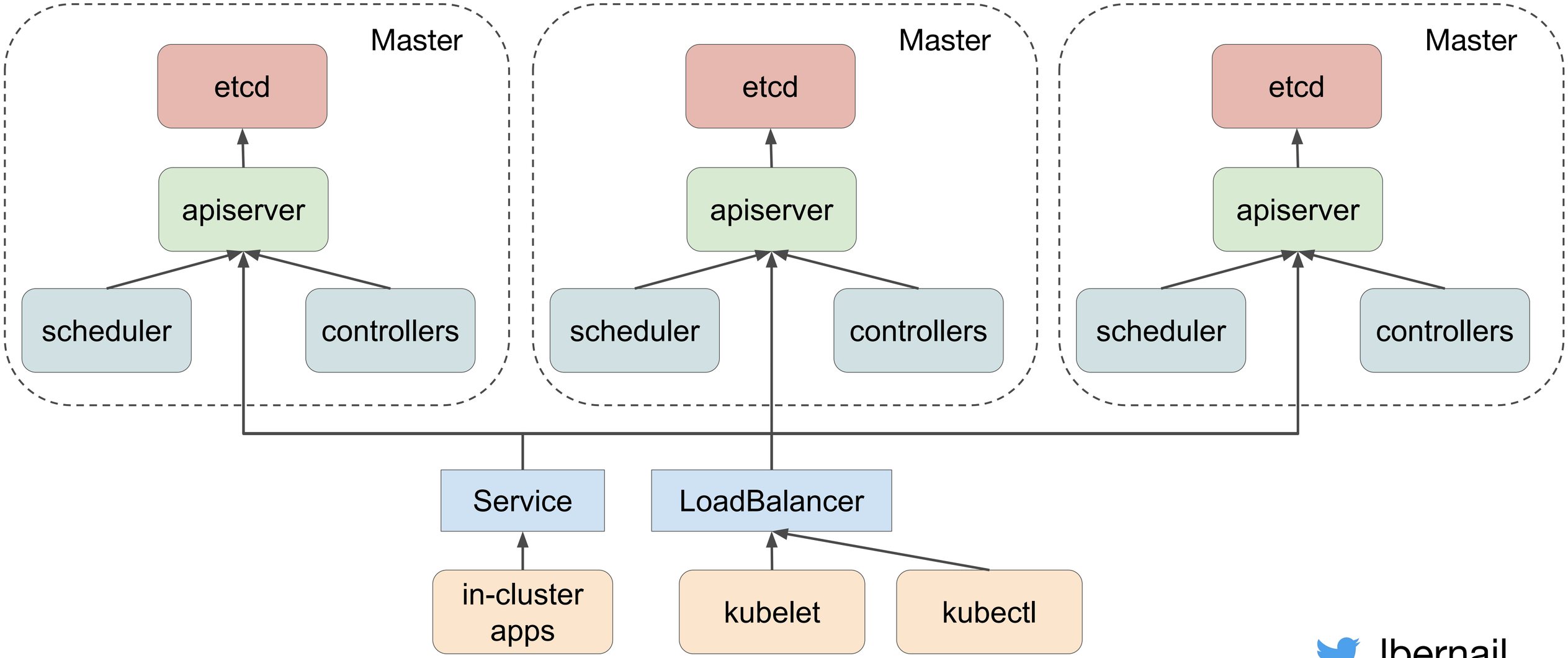
Ibernail

Resilient and Scalable Control Plane

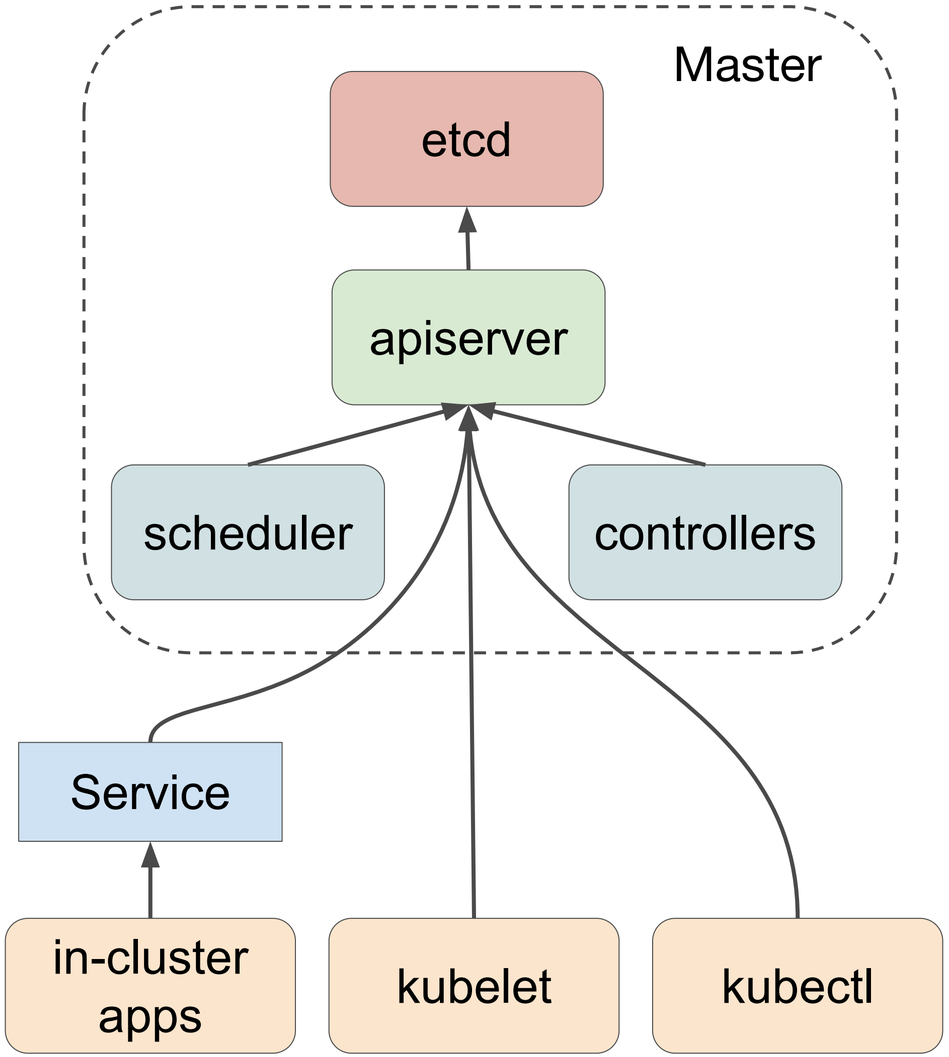
Kube 101 Control Plane



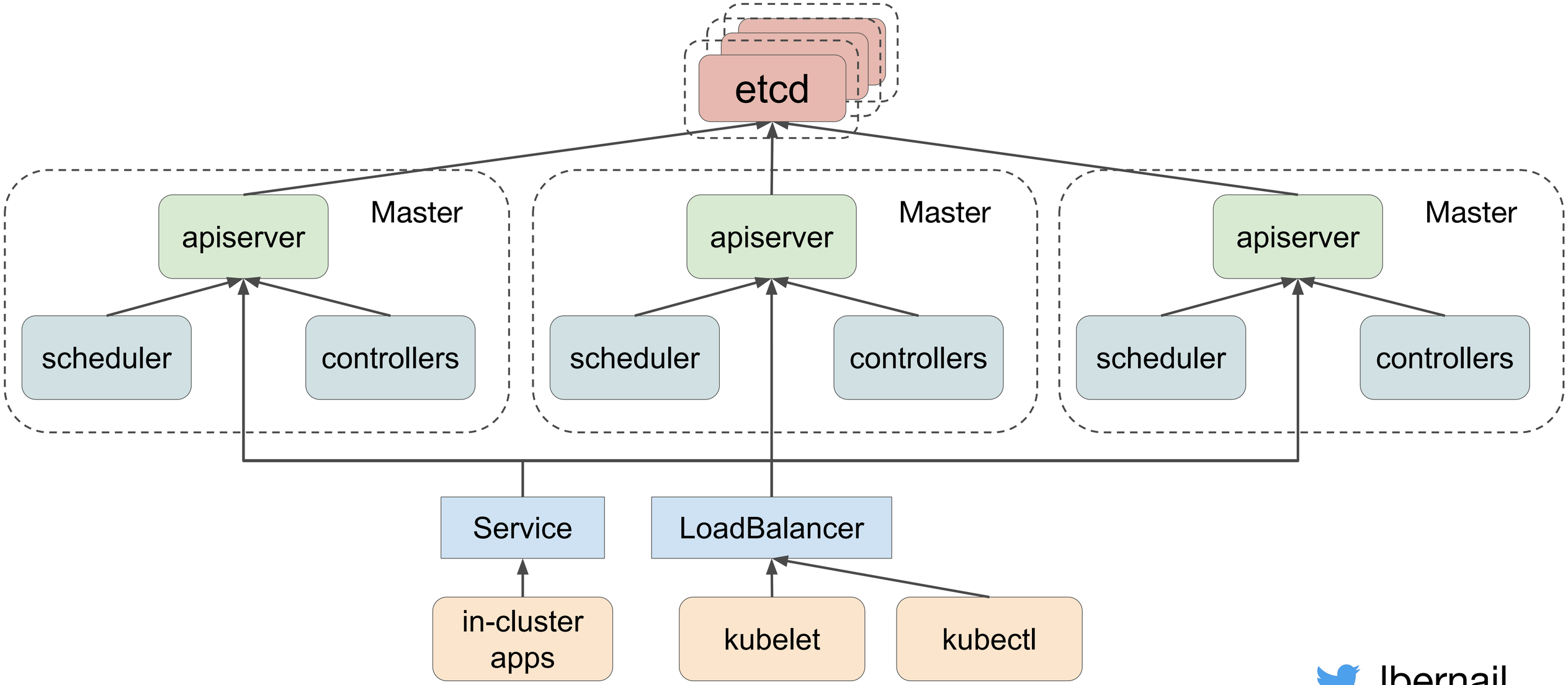
Making it resilient



Kube 101 Control Plane

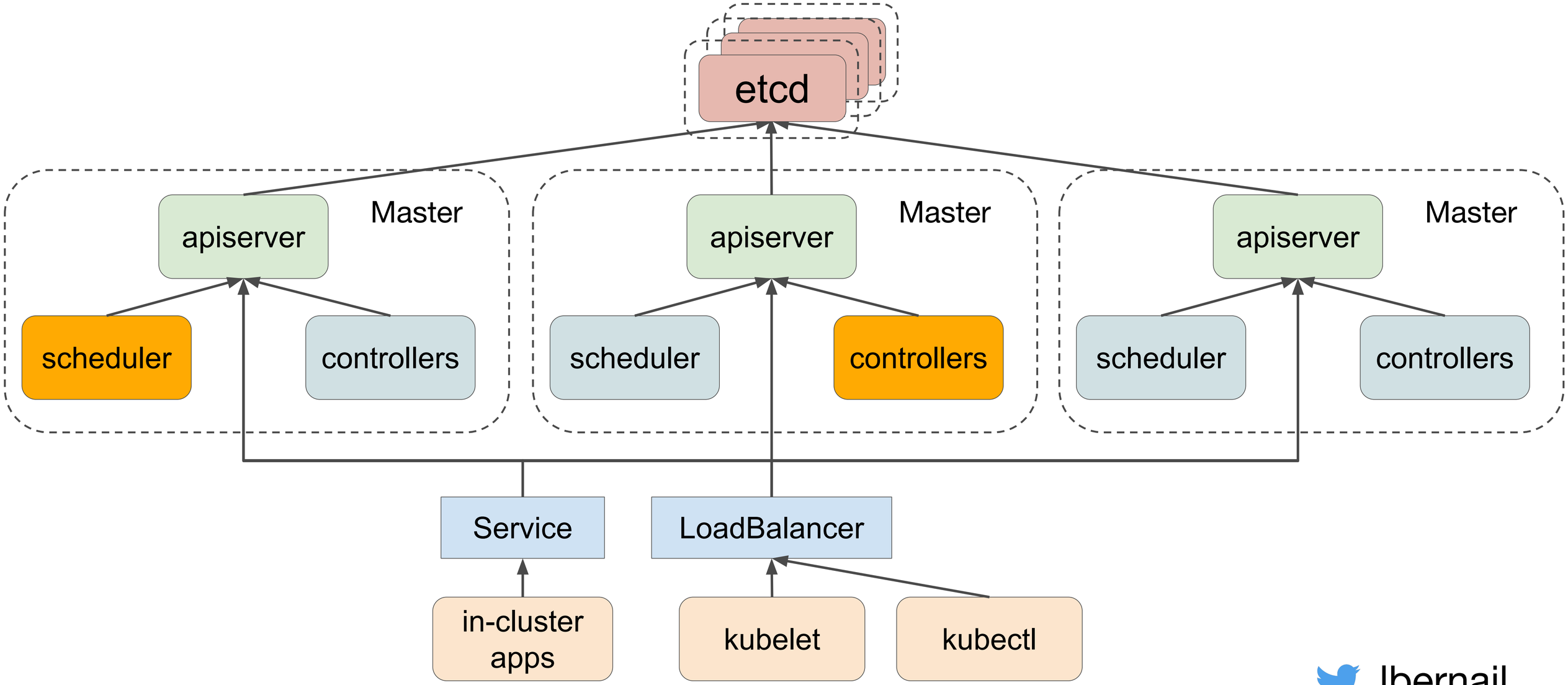


Separate etcd nodes



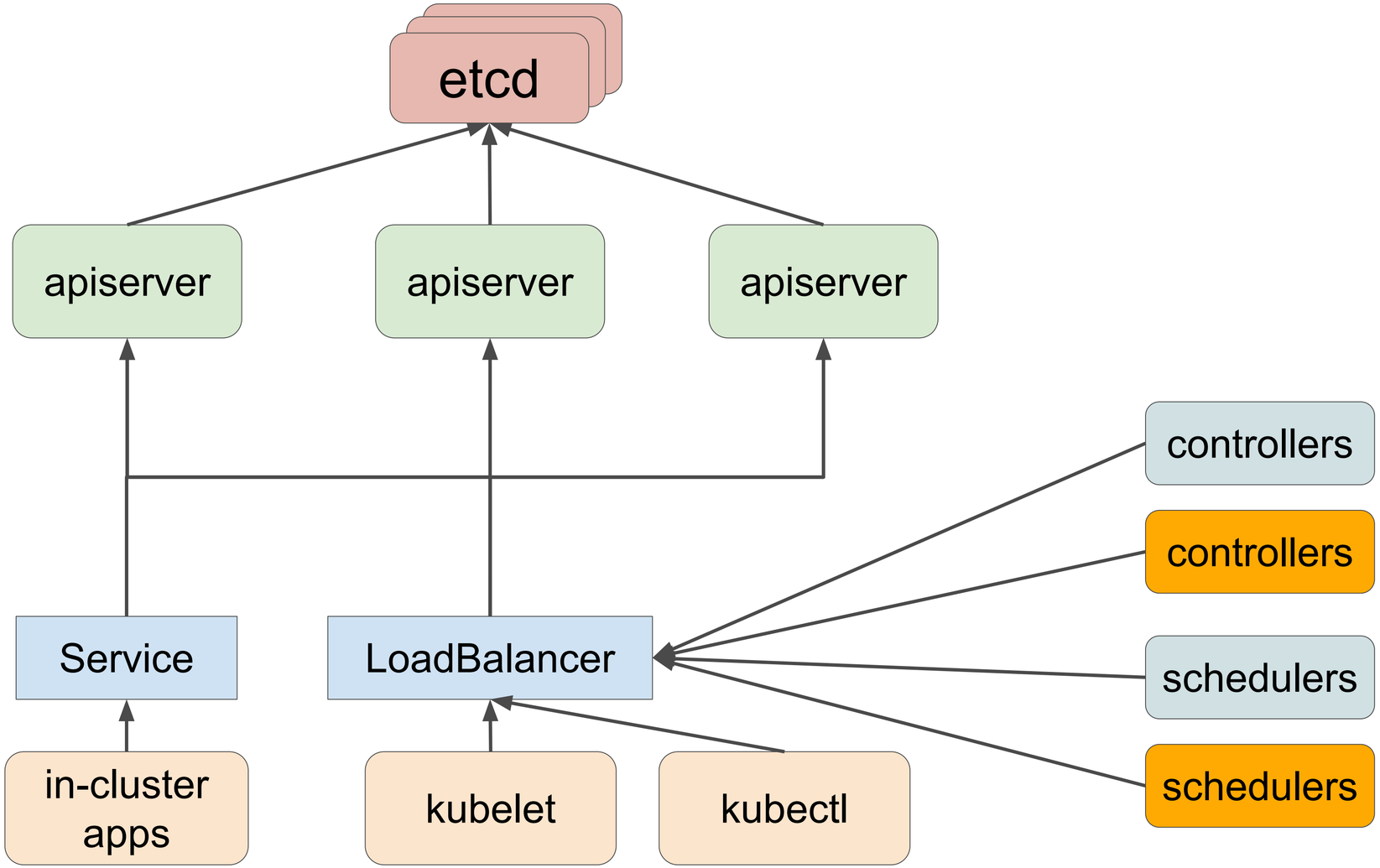
lbernail

Single active Controller/scheduler



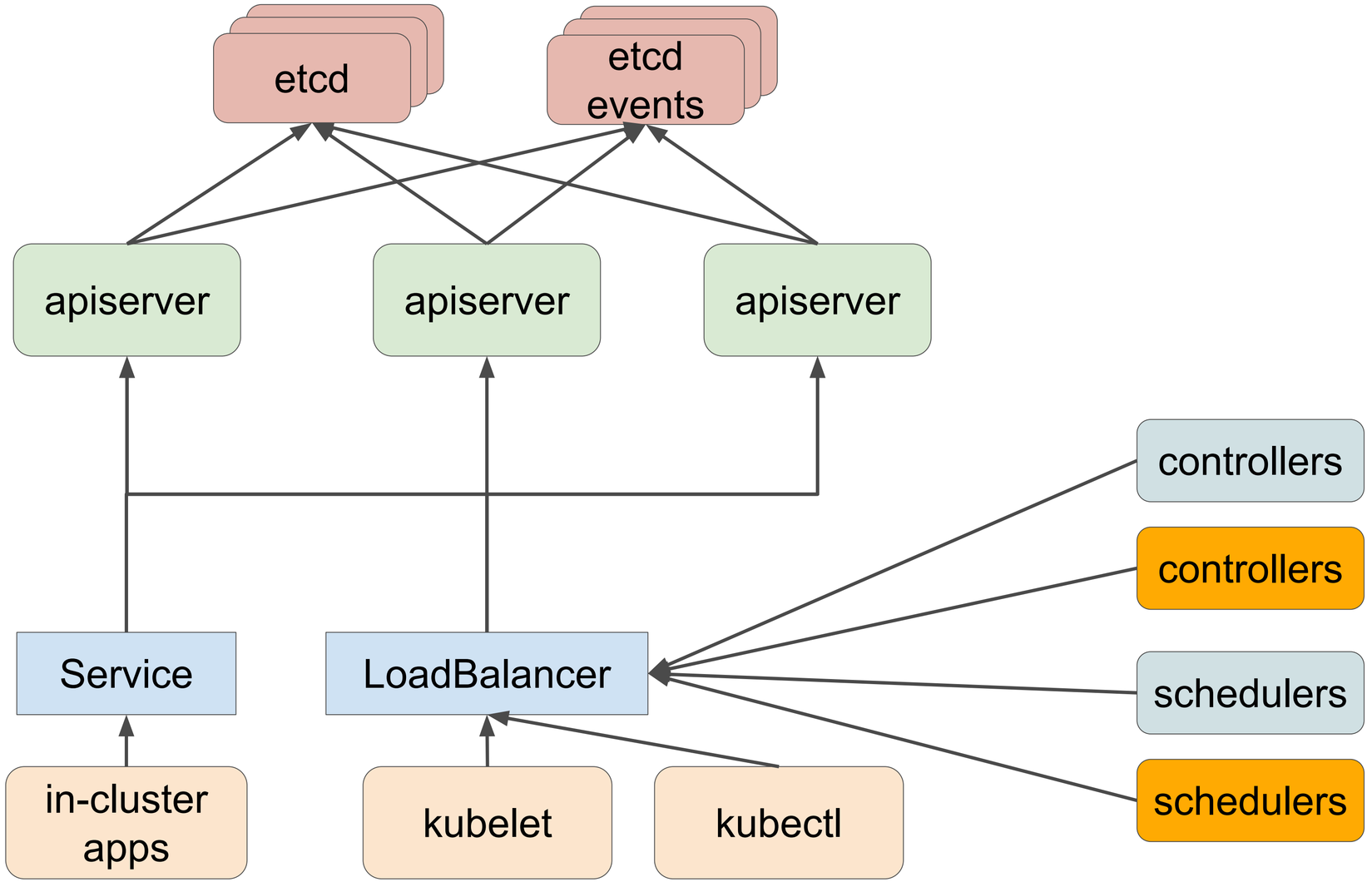
lbernail

Split scheduler/controllers



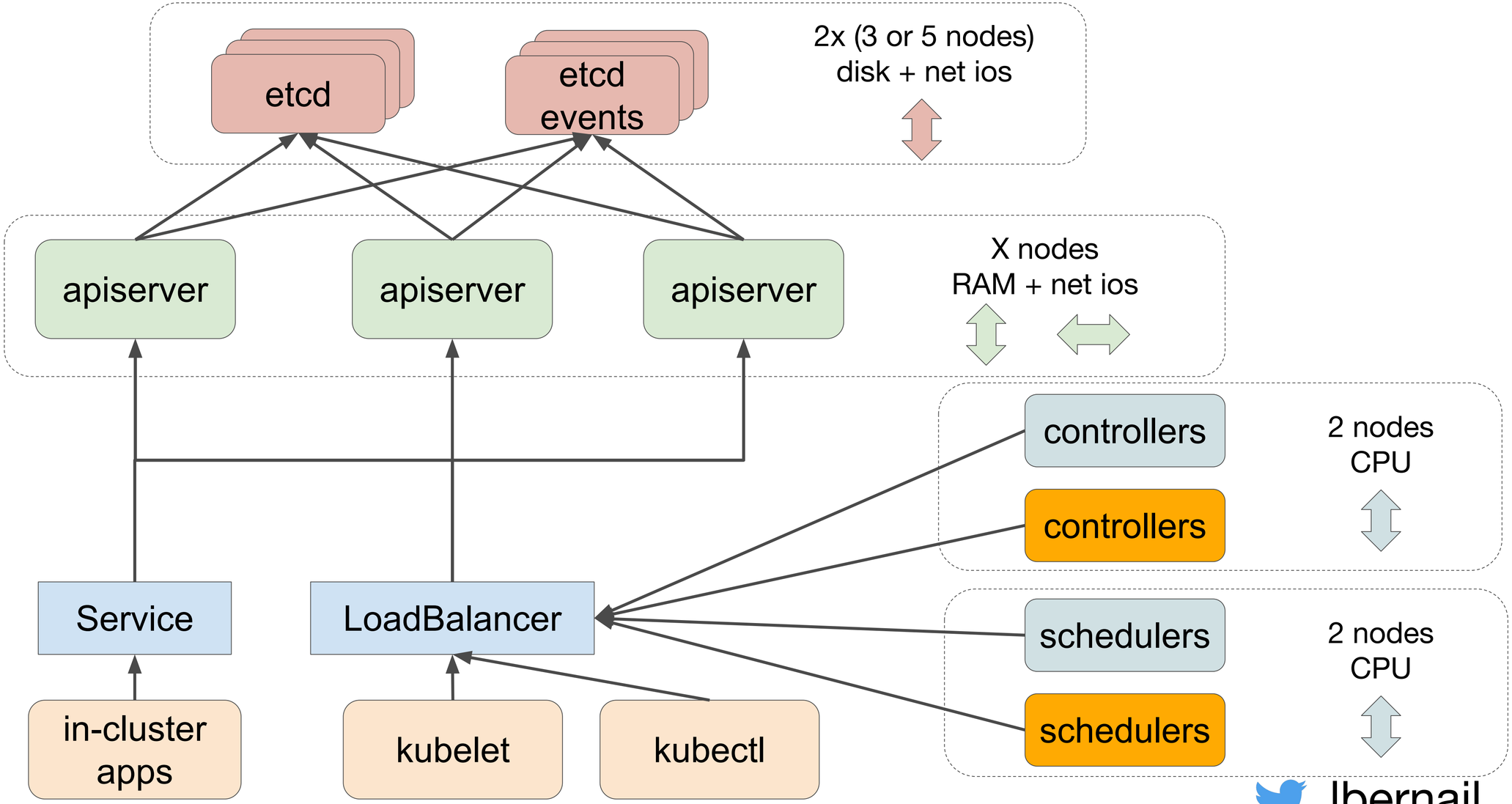
lbernail

Split etcd



lbernail

Sizing the control plane



What happens after “Kube 101”

1. Resilient and Scalable Control Plane
2. Securing the Control Plane
    a. Kubernetes and Certificates
    b. Exceptions?
    c. Impact of Certificate Rotation
3. Efficient networking
    a. Giving pod IPs and routing them
    b. Ingresses: Getting data in the cluster

Kubernetes and Certificates

From “the hard way”

```
cat > ca-config.json <<EOF
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": ["signing", "key encipherment", "server auth", "client auth"],
        "expiry": "8760h"
      }
    }
  }
}
```

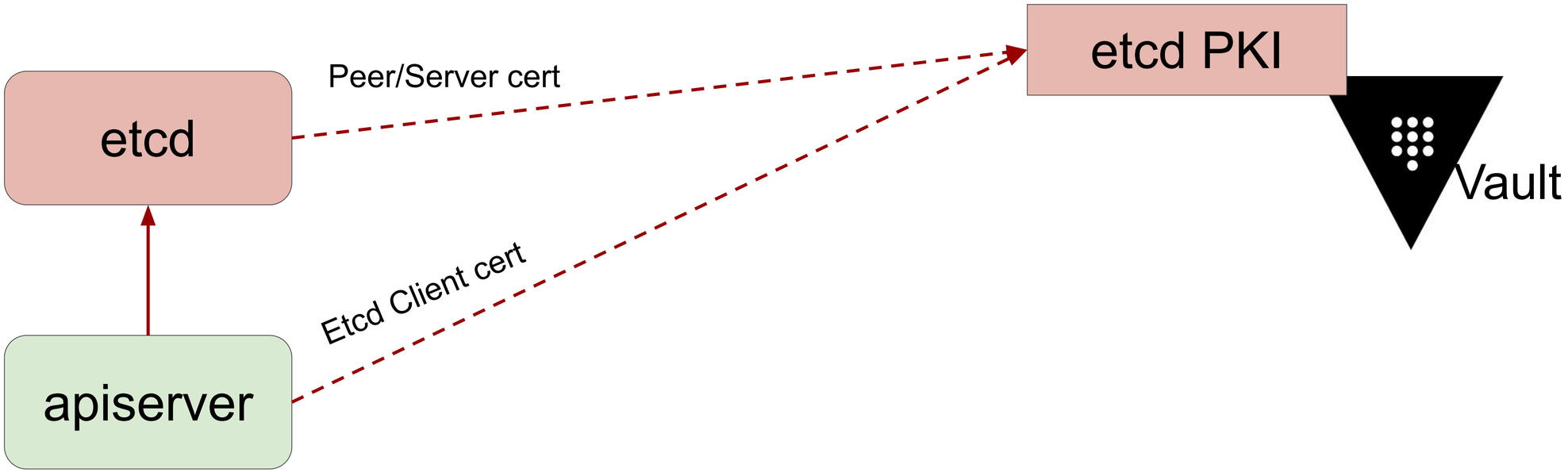
“Our cluster broke after ~1y”

```
cat > ca-config.json <<EOF
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": ["signing", "key encipherment", "server auth", "client auth"],
        "expiry": "8760h"
      }
    }
  }
}
```

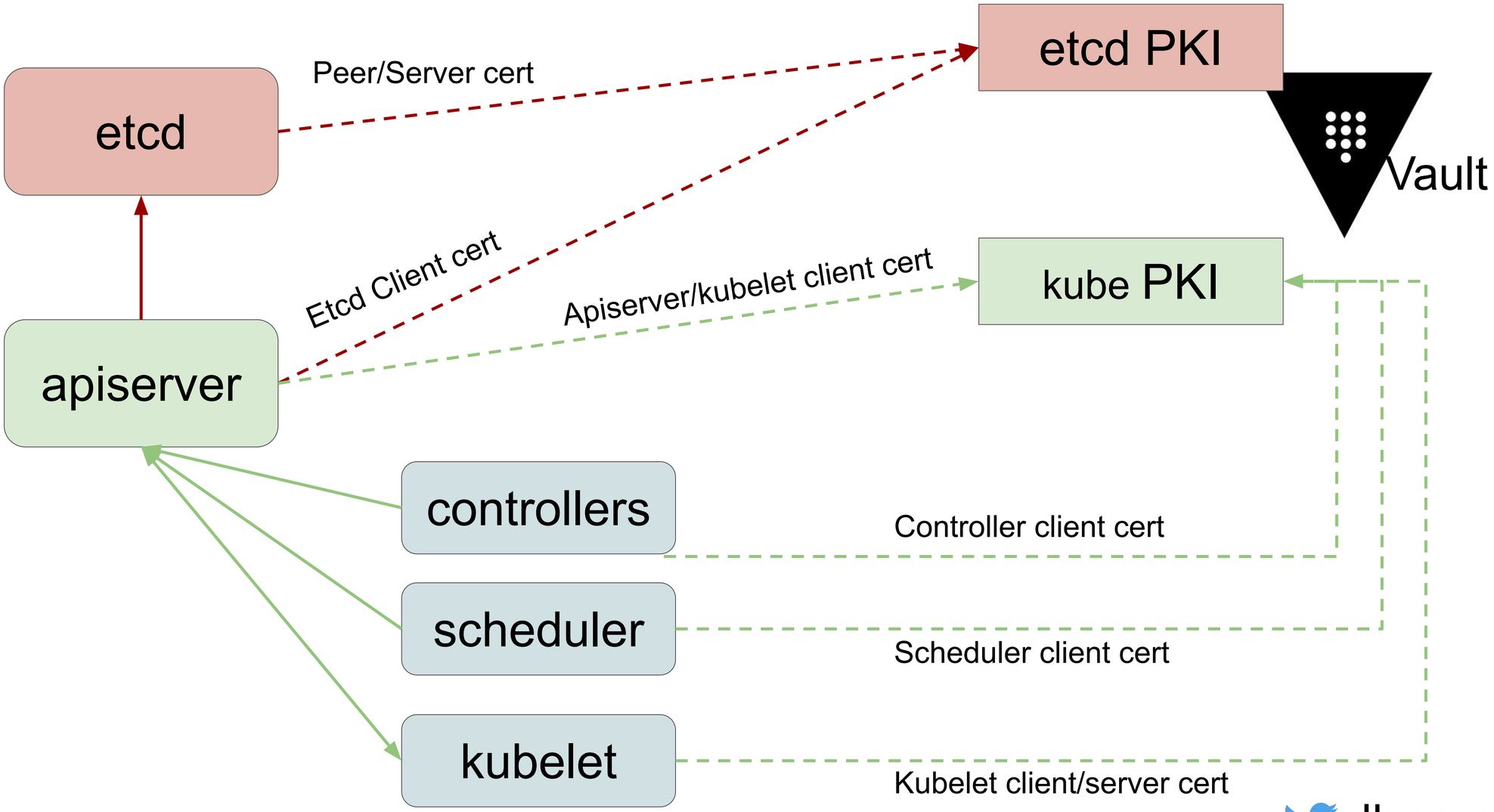
Certificates in Kubernetes

- Kubernetes uses certificates everywhere
- Very common source of incidents
- Our Strategy: Rotate all certificates daily

Certificate management

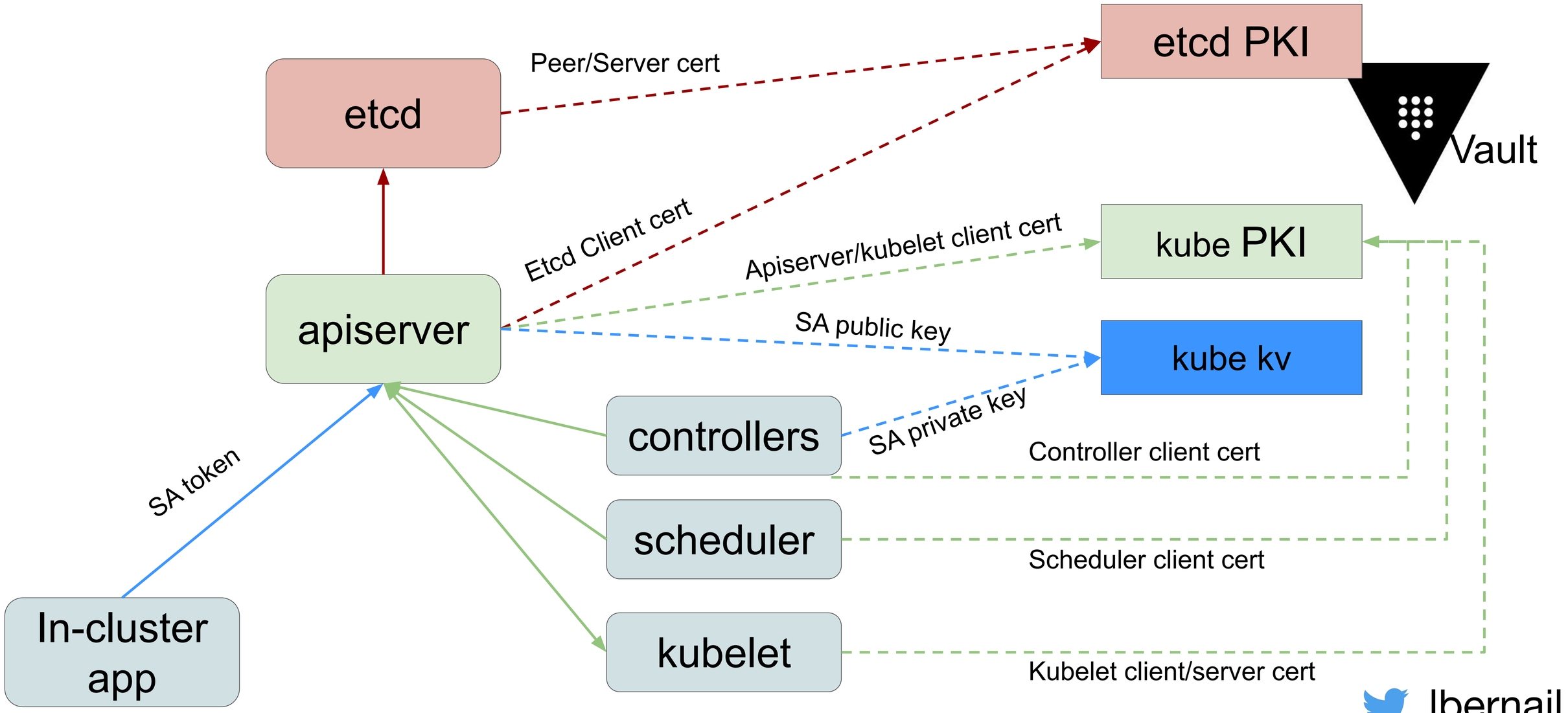


Certificate management

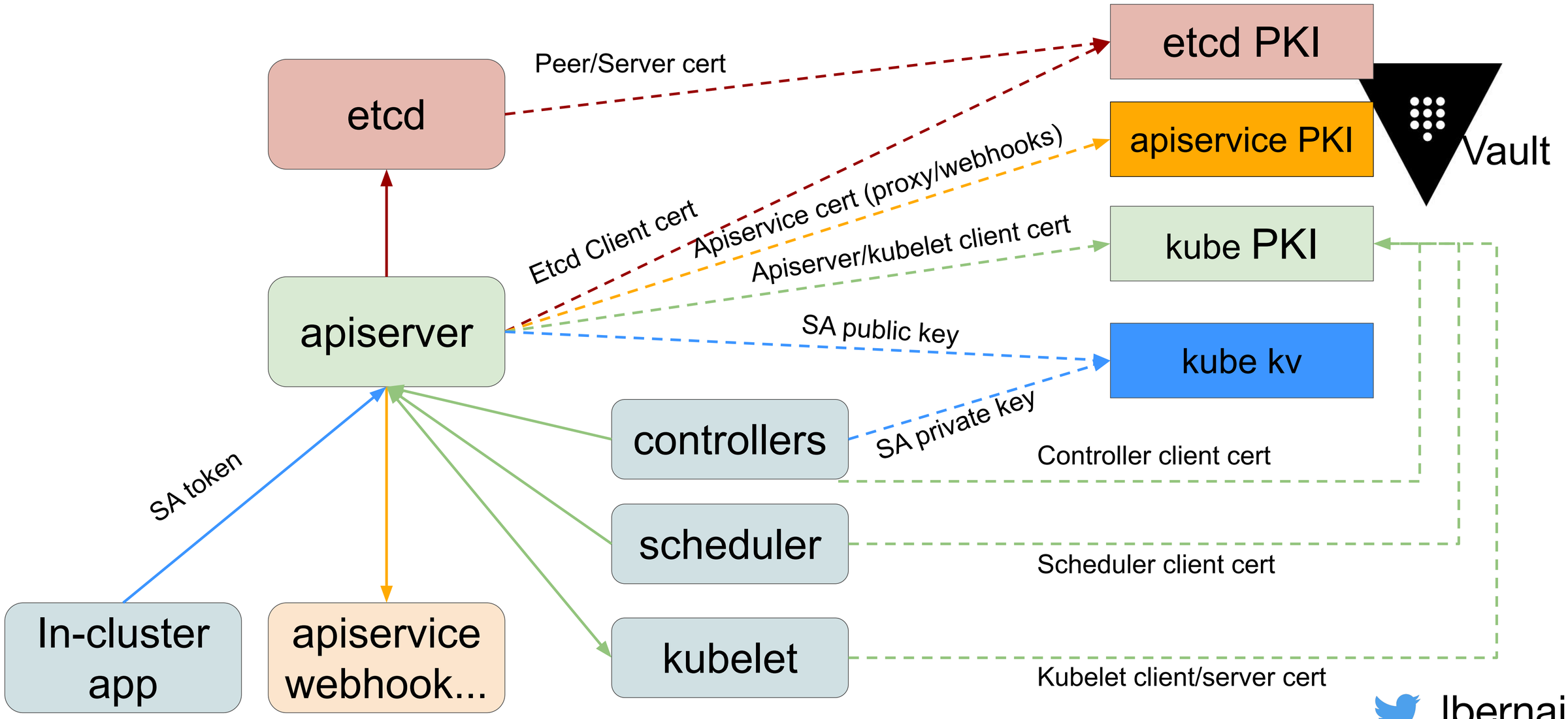


lbernail

Certificate management

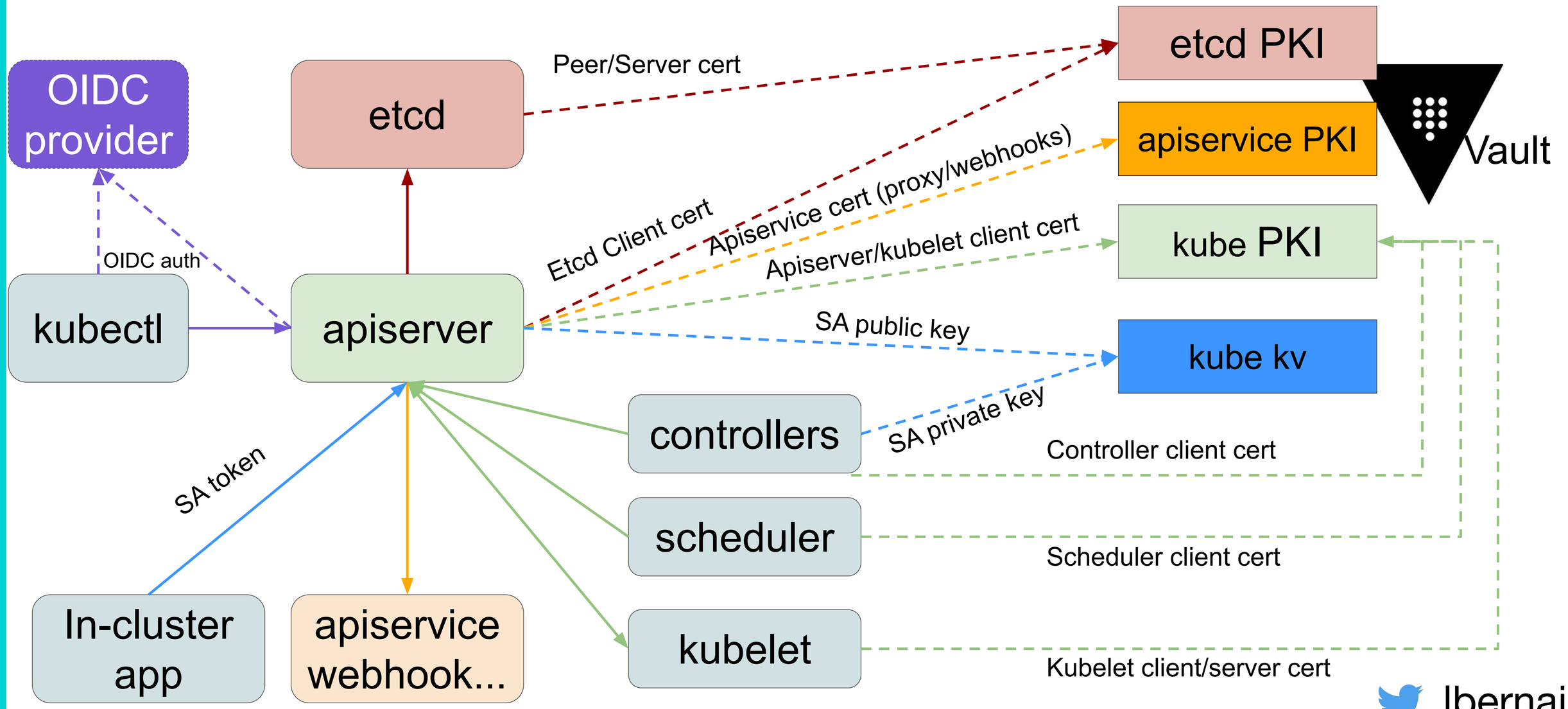


Certificate management



lbernail

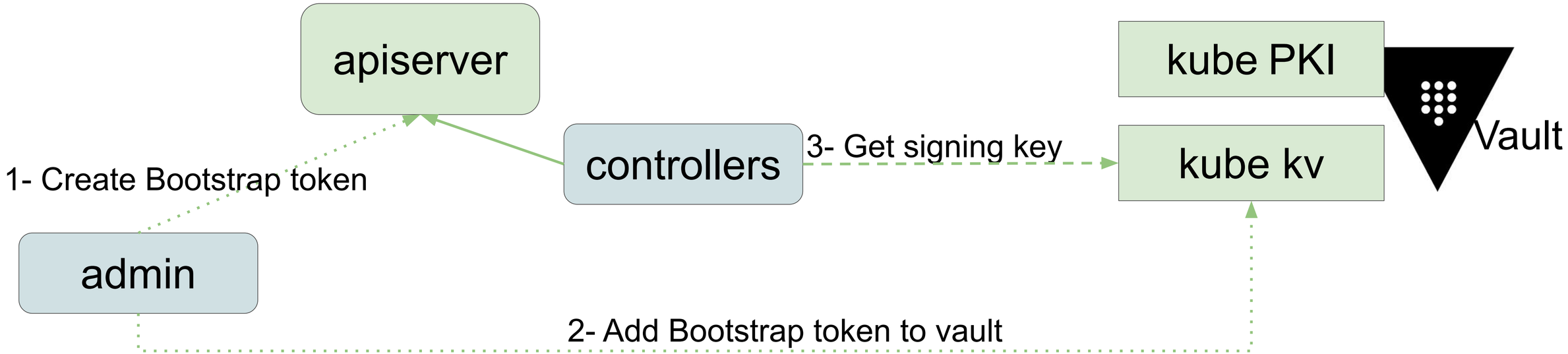
Certificate management



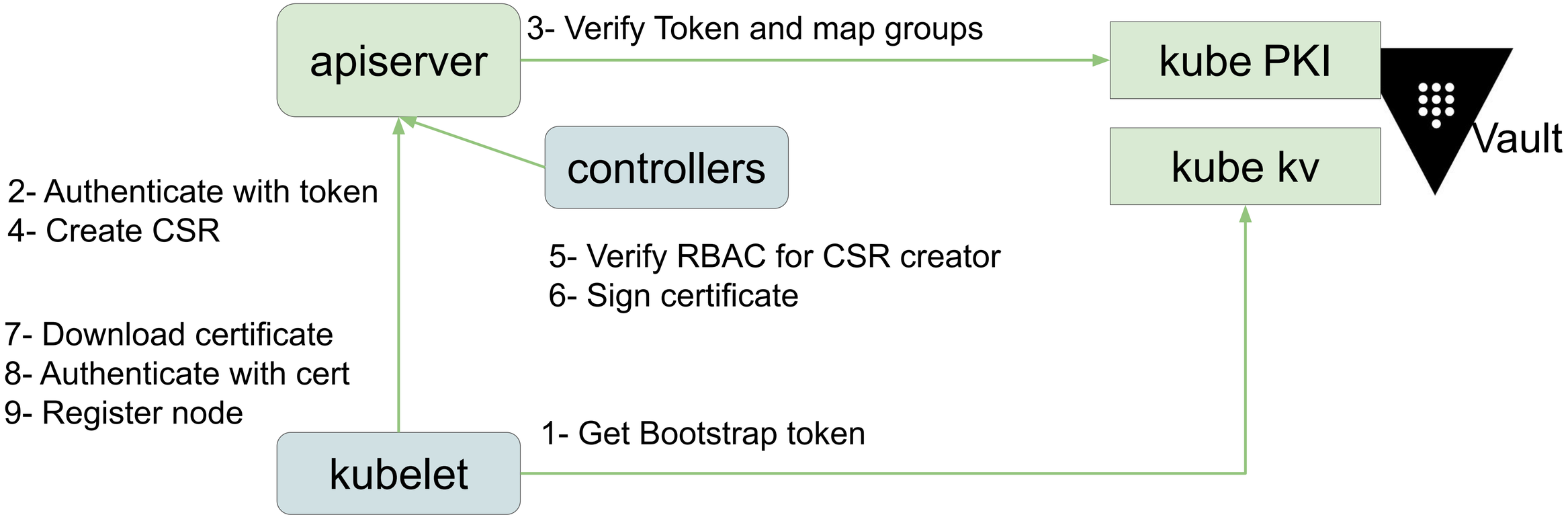
lbernail

Exception ?
Incident...

Kubelet: TLS Bootstrap



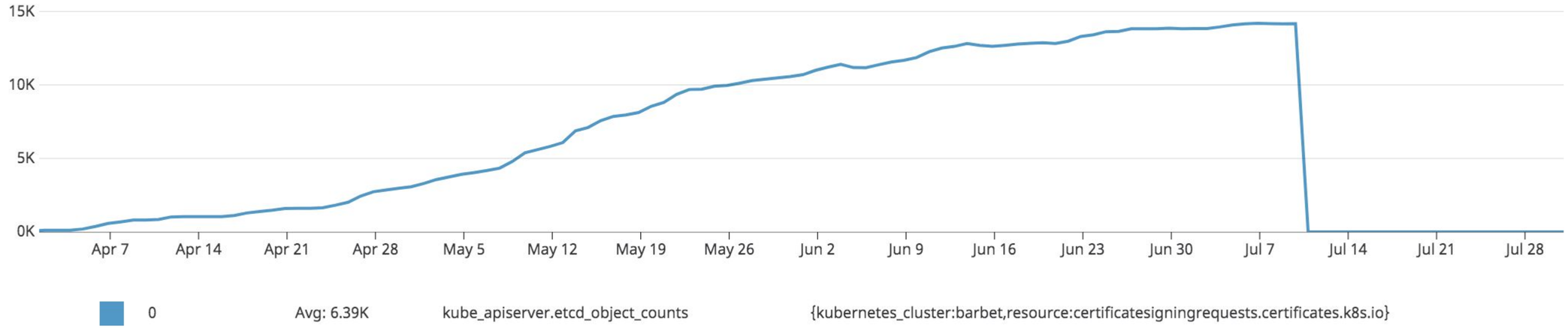
Kubelet: TLS Bootstrap



Kubelet certificate issue

1. One day, some Kubelets were failing to start or took 10s of minutes
2. Nothing in logs
3. Everything looked good but they could not get a cert
4. Turns out we had a lot of CSRs in flight
5. Signing controller was having a hard time evaluating them all

CSR resources in the cluster
Lower is better!



lbernail

Why?

Kubelet Authentication

- Initial creation: bootstrap token, mapped to group “system:bootstrappers”
- Renewal: use current node certificate, mapped to group “system:nodes”

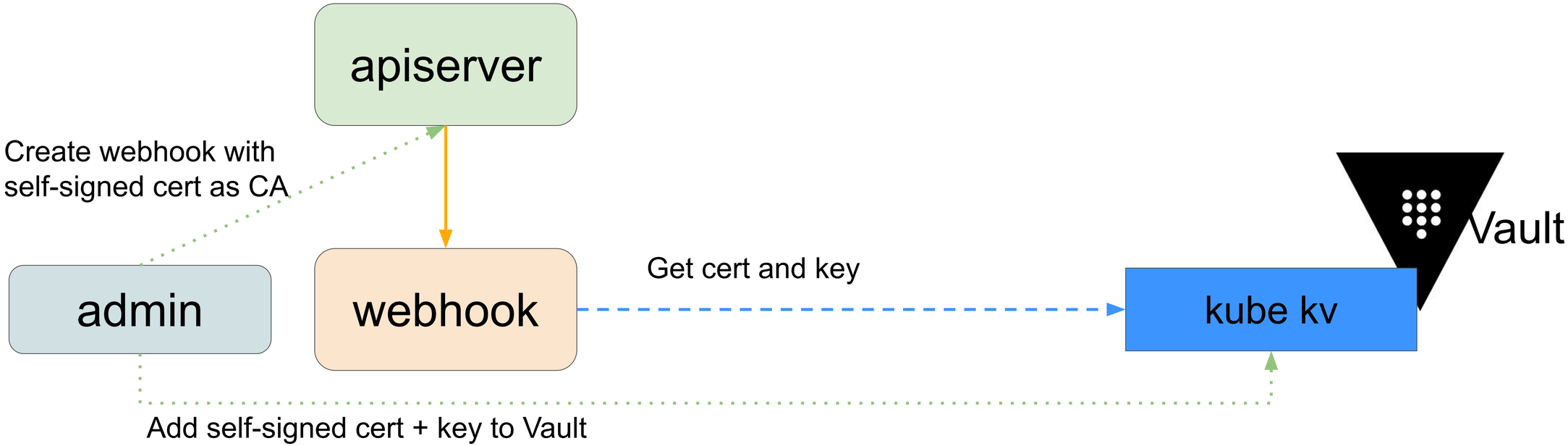
Required RBAC permissions

- CSR creation
- CSR auto-approval

	CSR creation	CSR auto-approval
system:bootstrappers	OK	OK
system:nodes	OK	

Exception 2?
Incident 2...

Temporary solution



One day, after ~1 year

- Creation of resources started failing (luckily only a Custom Resource)
- Cert had expired...

Take-away

- Rotate server/client certificates
- Not easy

But, “If it’s hard, do it often”

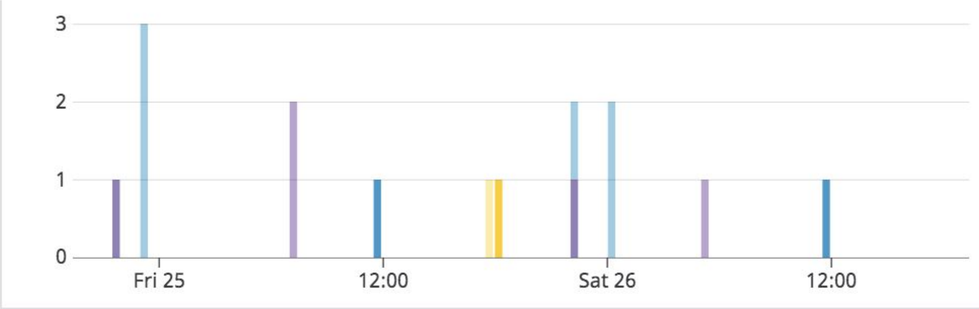
> *no expiration issues anymore*

Impact of Certificate rotation

Apiserver certificate rotation

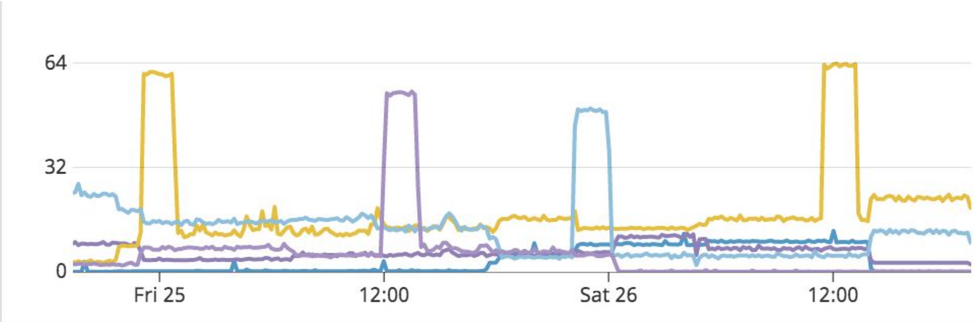
Impact on etcd

apiserver restarts



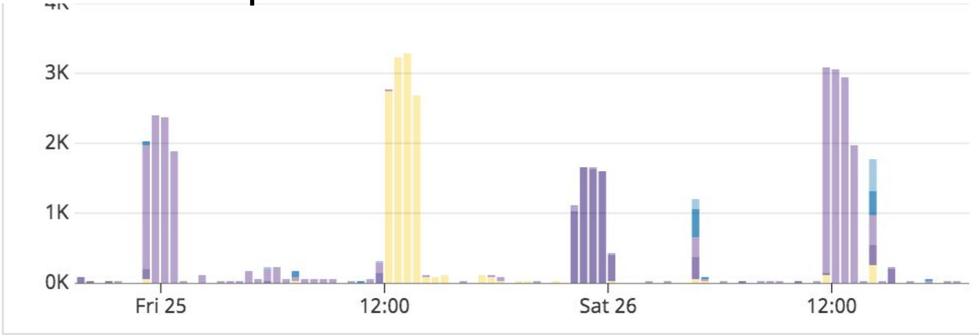
We have multiple apiservers
We restart each daily

etcd traffic



Significant etcd network impact
(caches are repopulated)

etcd slow queries

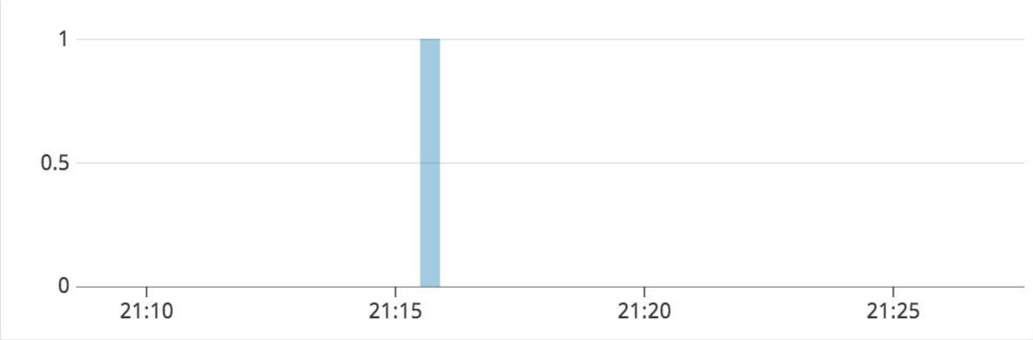


Significant impact on etcd performances

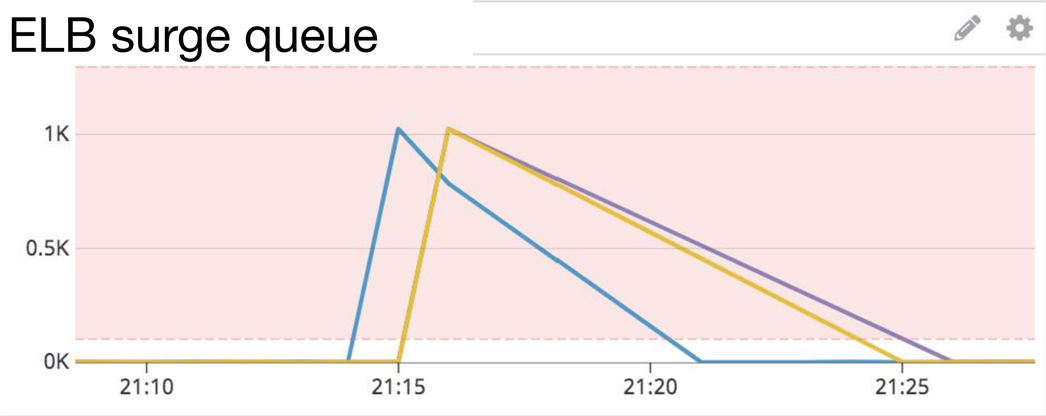
Ibernail

Impact on Load-balancers

apiserver restarts



ELB surge queue



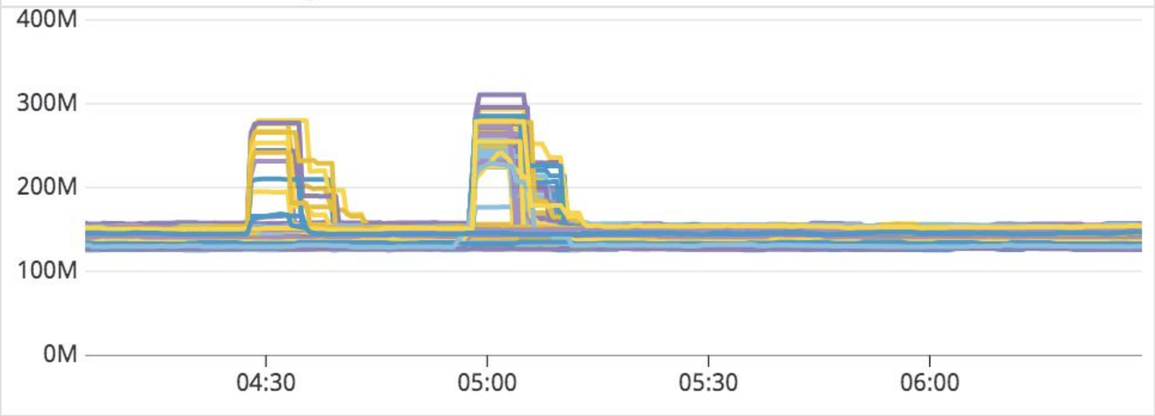
Significant impact on LB as connections are reestablished

Mitigation: increase queues on apiservers
net.ipv4.tcp_max_syn_backlog
net.core.somaxconn

Impact on apiserver clients

apiserver restarts

coredns memory usage



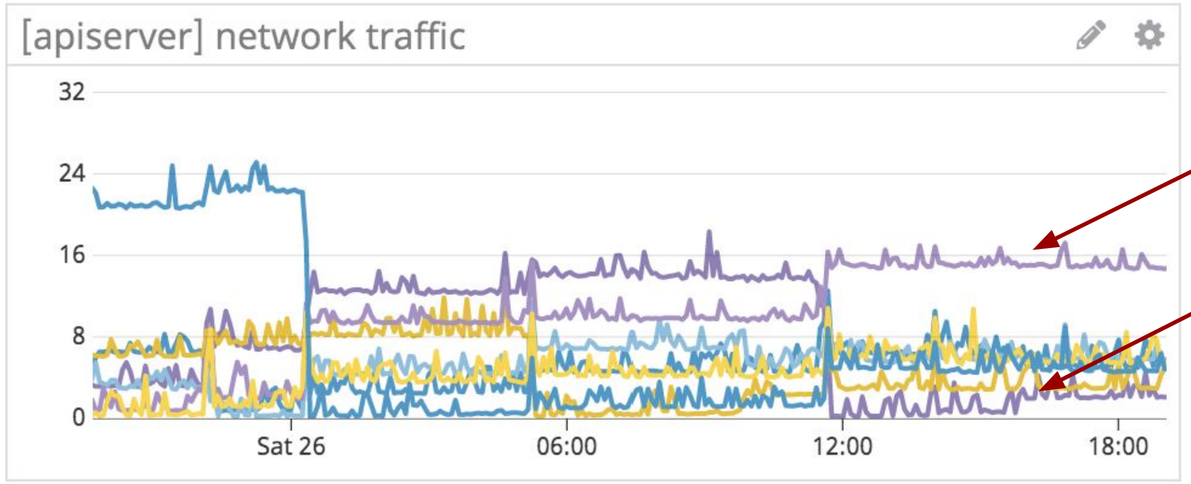
- Apiserver restarts
- clients reconnect and refresh their cache

> Memory spike for impacted apps

No real mitigation today

lbernail

Impact on traffic balance

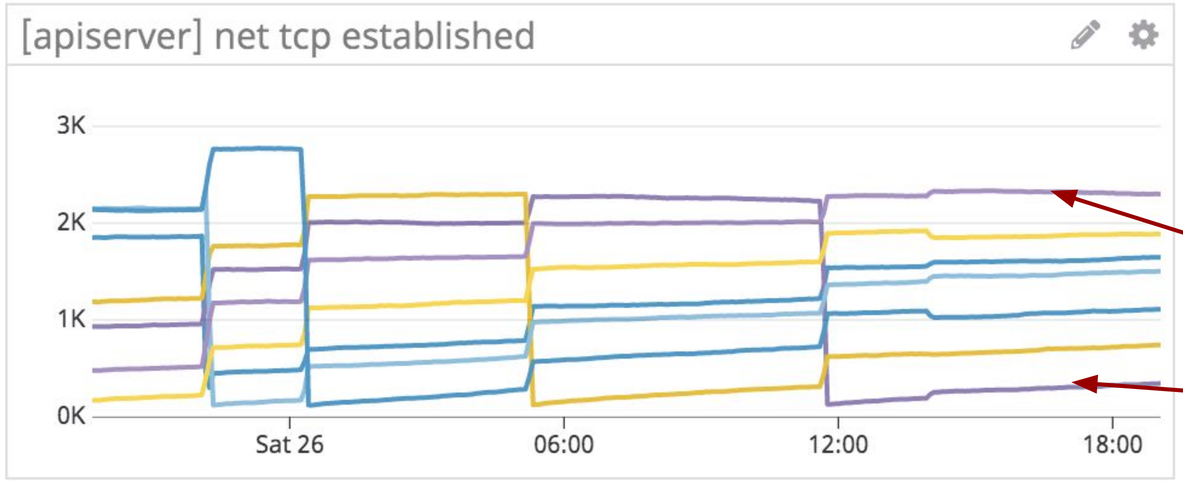


15MB/s

2.5MB/s

Number of connections / traffic very unbalanced
Because connections are very long-lived

More clients => Bigger impact clusterwide

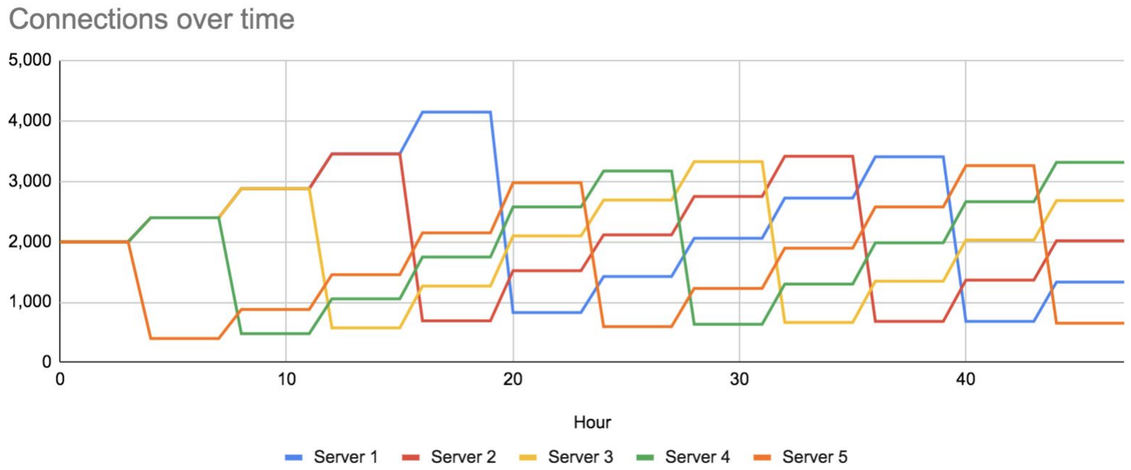


2300 connections

300 connections

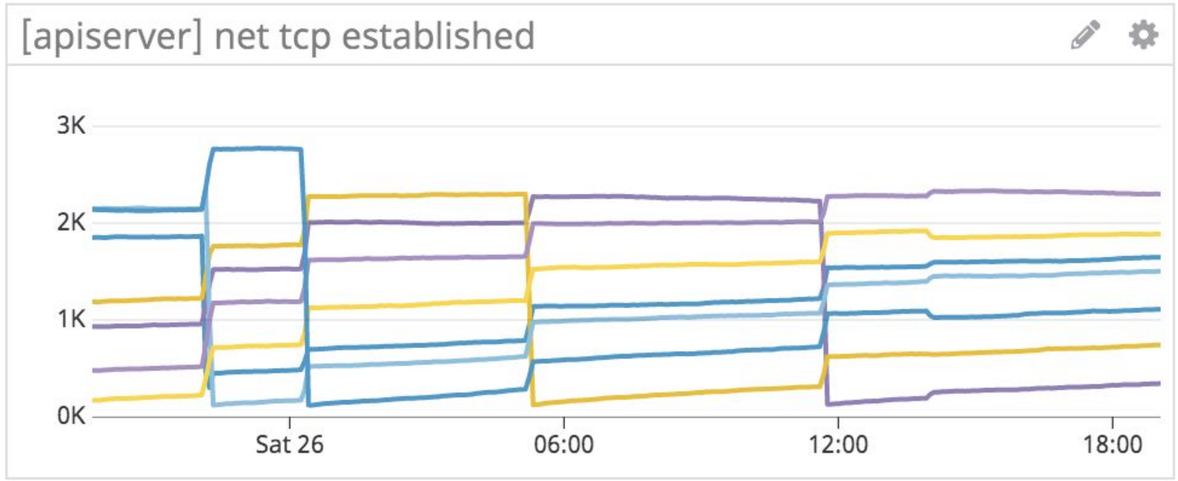
Ibernail

Why? Simple simulation



Simulation for 48h

- 5 apiservers
- 10000 connections (4 x 2500 nodes)
- Every 4h, one apiserver restarts
- Reconnections evenly dispatched

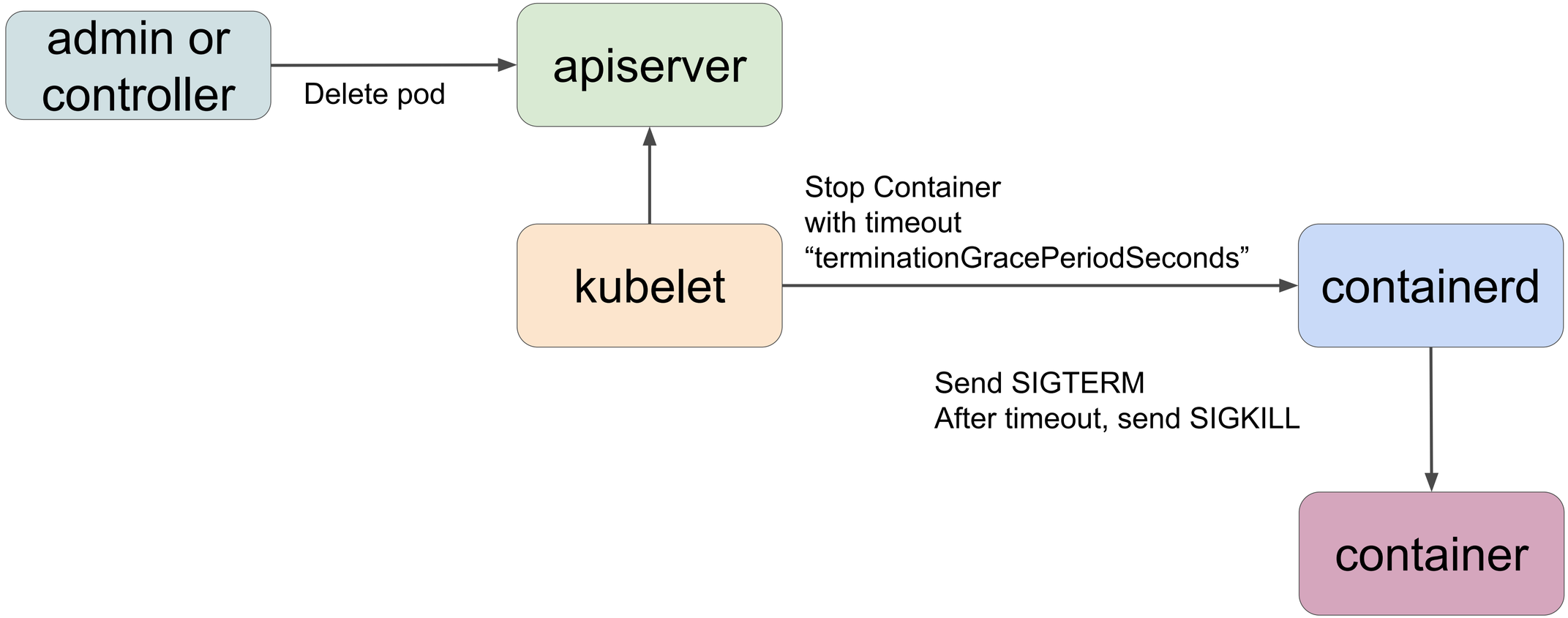


Cause

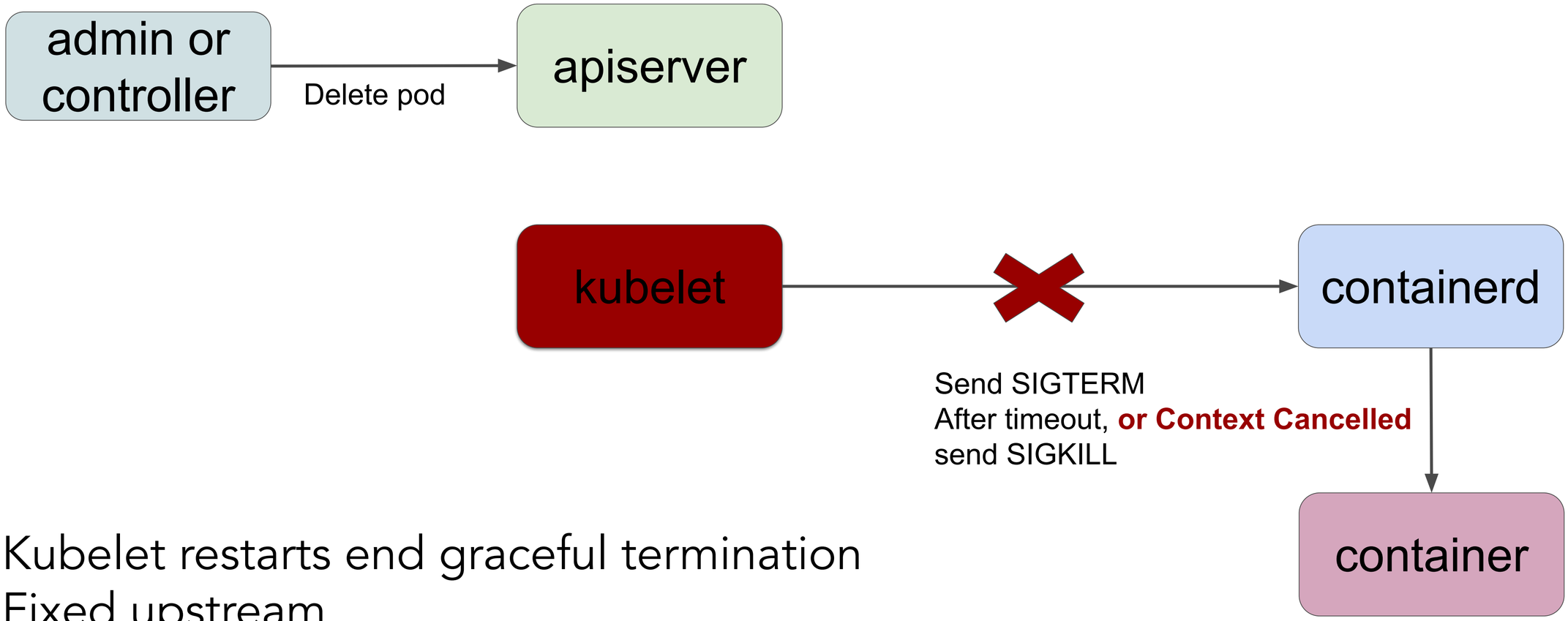
- Cloud TCP load-balancers use round-robin
- Long-lived connections
- No rebalancing

Kubelet certificate rotation

Pod graceful termination



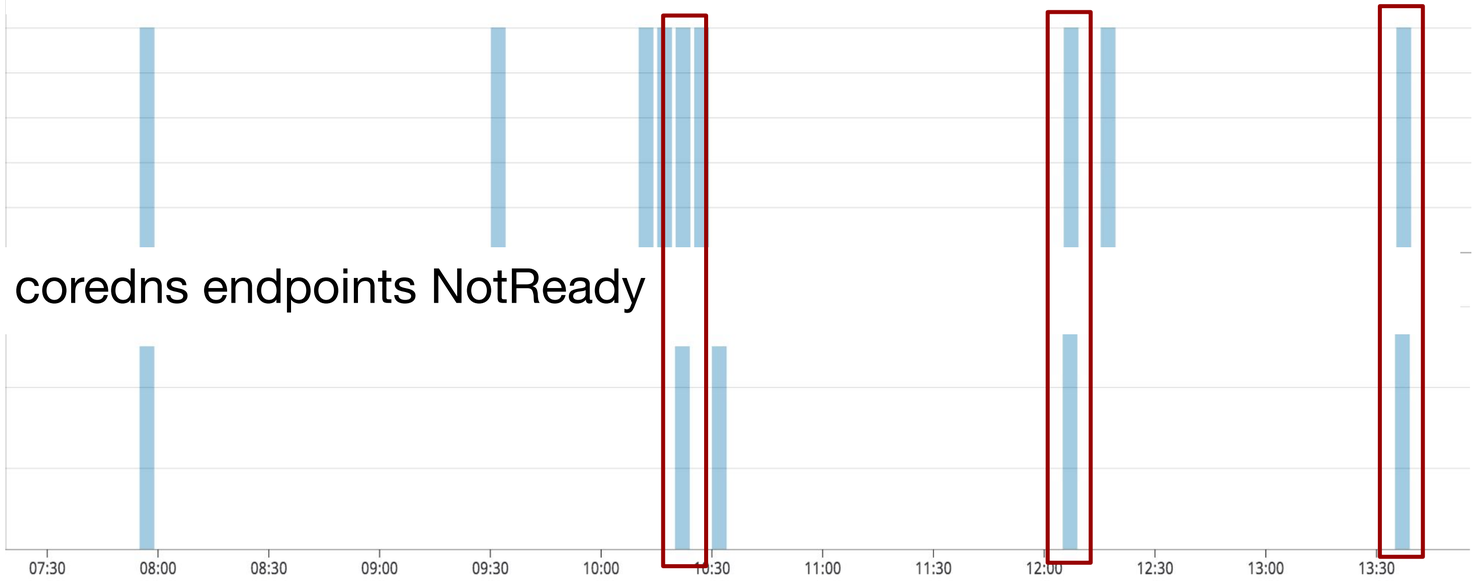
Restarts impact graceful termination



Kubelet restarts end graceful termination
Fixed upstream
“Do not SIGKILL container if container stop is cancelled”
https://github.com/containerd/cri/pull/1099

Impact on pod readiness

kubelet restarts on “system” nodes (coredns + other services)



On kubelet restart

- Readiness probes marked as failed
- Pods removed from service endpoints
- Requires readiness to succeed again

Issue upstream

“pod with readinessProbe will be not ready when kubelet restart”
https://github.com/kubernetes/kubernetes/issues/78733

Take-away

Restarting components is not transparent

It would be great if

- Components could transparently reload certs (server & client)
- Clients could wait 0-Xs to reconnect to avoid thundering herd
- Reconnections did not trigger memory spikes
- Cloud TCP load-balancers supported least-conn algorithm
- Connections were rebalanced (kill them after a while?)

What happens after “Kube 101”

1. Resilient and Scalable Control Plane
2. Securing the Control Plane
    a. Kubernetes and Certificates
    b. Exceptions?
    c. Impact of Certificate Rotation
3. Efficient networking
    a. Giving pod IPs and routing them
    b. Ingresses: Getting data in the cluster

Efficient networking

Network challenges

Throughput

Trillions of data points daily

Scale

1000-2000 nodes clusters

Latency

End-to-end pipeline

Topology

Multiple clusters

Access from standard VMs

Giving pods IPs & Routing them

From “the Hard Way”

Routes

Create network routes for each worker instance:

```
for i in 0 1 2; do
  gcloud compute routes create kubernetes-route-10-200-${i}-0-24 \
    --network kubernetes-the-hard-way \
    --next-hop-address 10.240.0.2${i} \
    --destination-range 10.200.${i}.0/24
done
```

node IP

Pod CIDR for this node

Small cluster? Static routes

Node 1

IP: 192.168.0.1
Pod CIDR: 10.0.1.0/24

Node 2

IP: 192.168.0.2
Pod CIDR: 10.0.2.0/24

Routes (local or cloud provider)
10.0.1.0/24 => 192.168.0.1
10.0.2.0/24 => 192.168.0.2

Limits
local: nodes must be in the same subnet
cloud provider: number of routes

Mid-size cluster? Overlay

Limits

- Overhead of the overlay
- Scaling route distribution (control plane)

Large cluster with a lot of traffic? Native pod routing

Performance

Datapath: no overhead
Control plane: simpler

Addressing

Pod IPs are accessible from

- Other clusters
- VMs

In practice

On premise

BGP
- Calico
- Kube-router

Macvlan

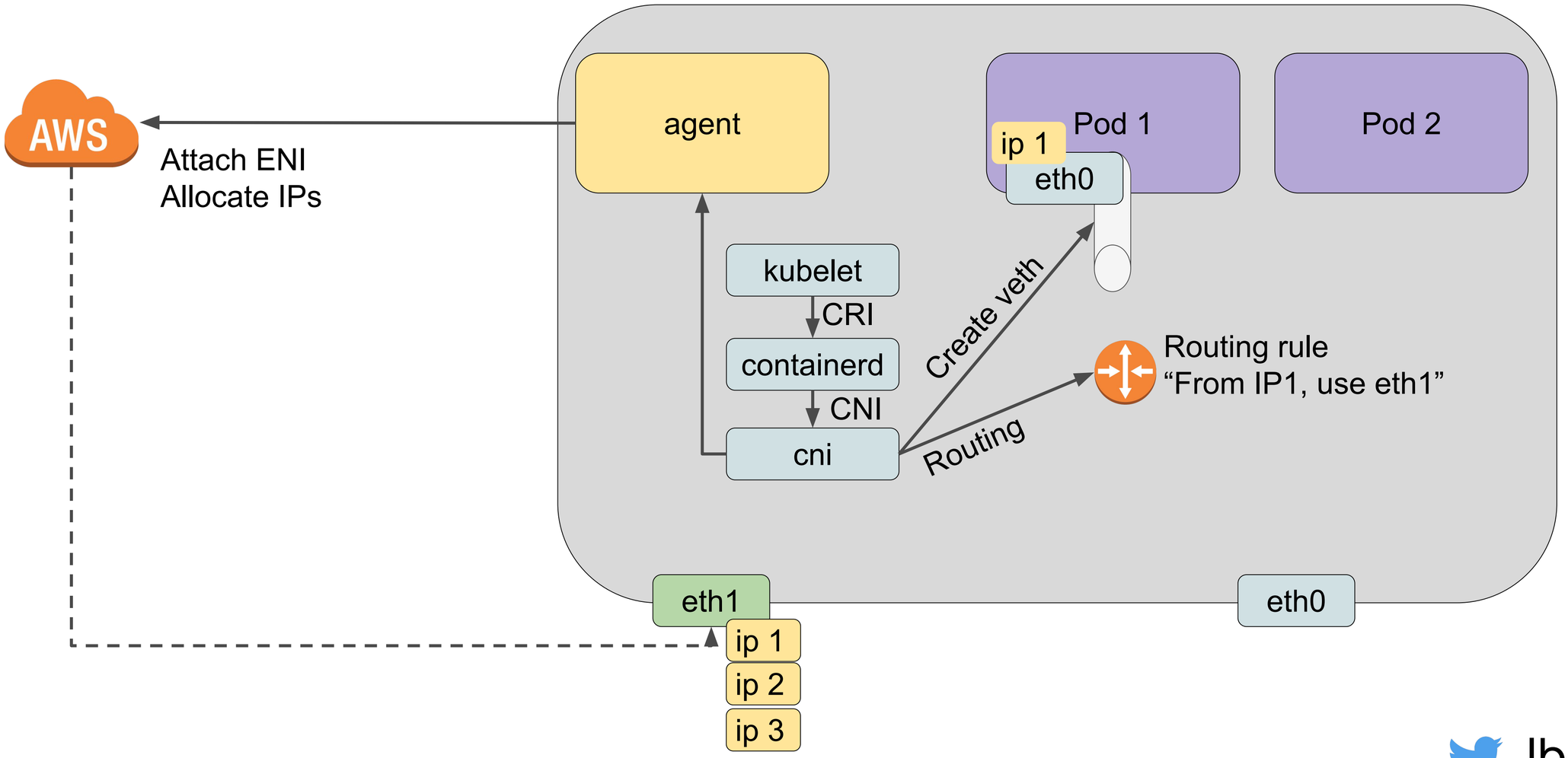
GCP

IP aliases

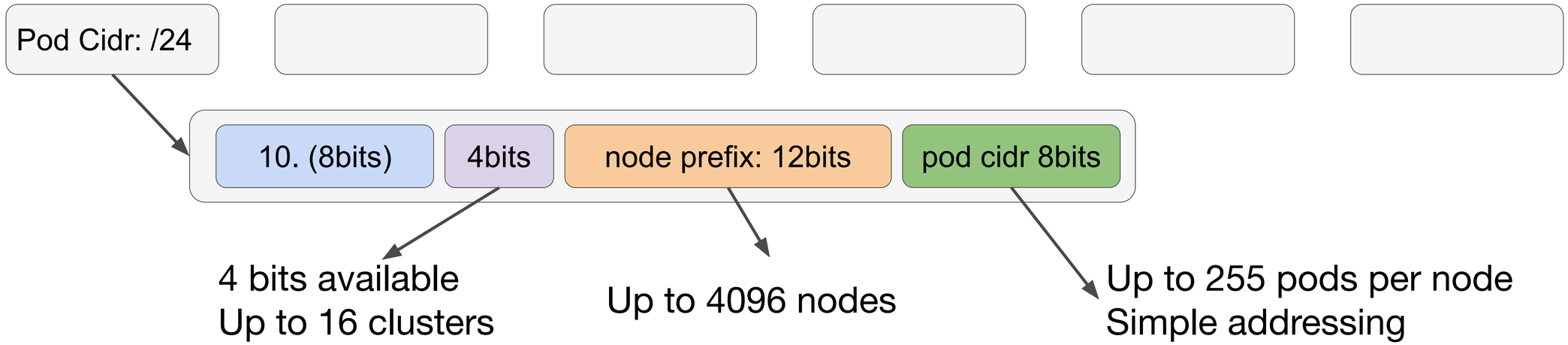
AWS

Additional IPs on ENIs
- AWS EKS CNI plugin
- Lyft CNI plugin
- Cilium ENI IPAM

How it works on AWS



Address space planning



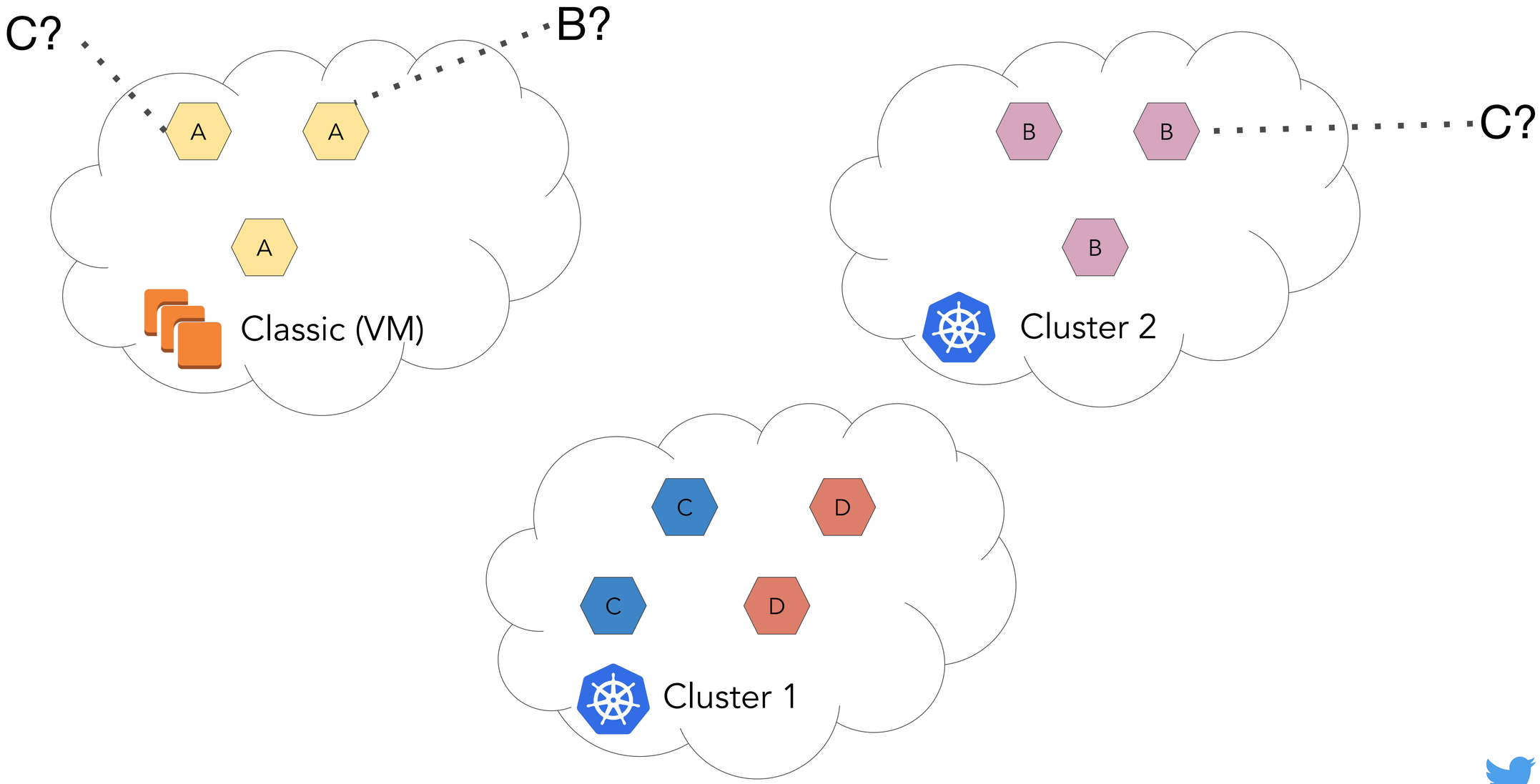
- /24 leads to inefficient address usage
- sig-network: remove contiguous range requirement for CIDR allocation
- But also
  - Address space for node IPs (another /20 per cluster for 4096 nodes)
  - Service IP range (/20 would make sense for such a cluster)
- Total: 1 /15 for pods, 2 /20 for nodes and service!

Take-away

- Native pod routing has worked very well at scale
- A bit more complex to debug
- Much more efficient datapath
- Topic is still dynamic (Cilium introduced ENI recently)
- Great relationship with Lyft / Cilium
- Plan your address space early

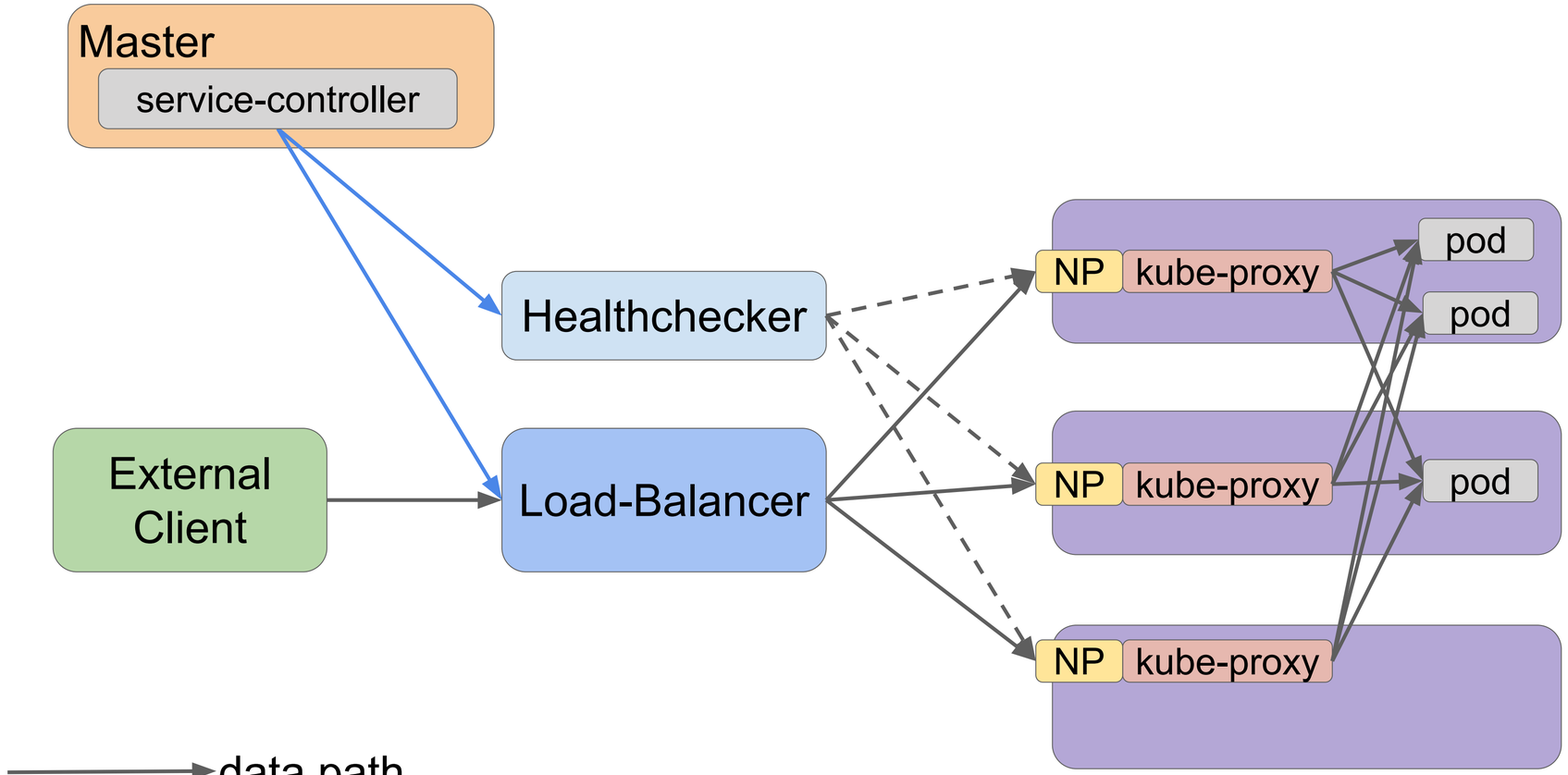
Ingresses

Ingress: cross-clusters, VM to clusters



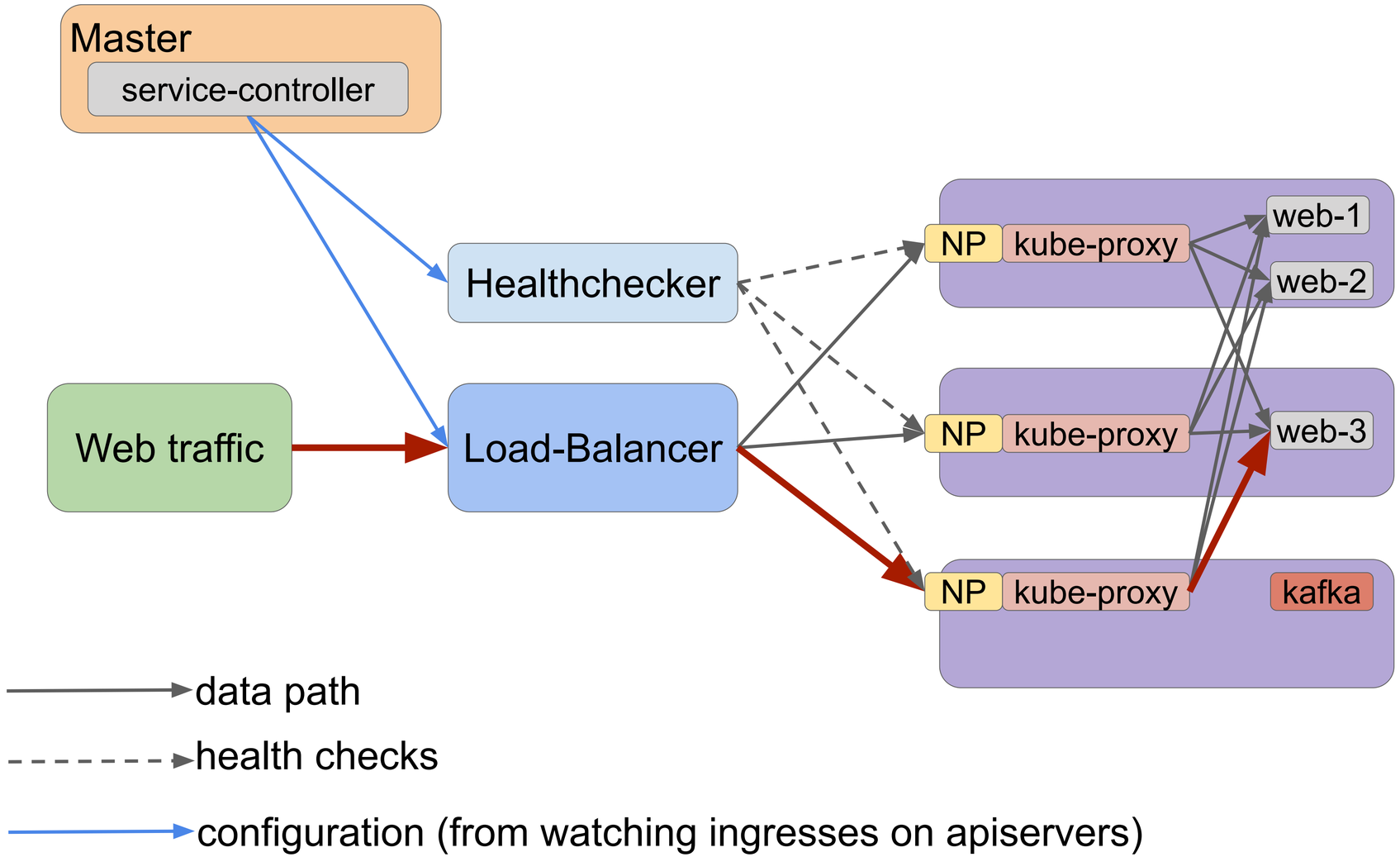
lbernail

Kubernetes default: LB service



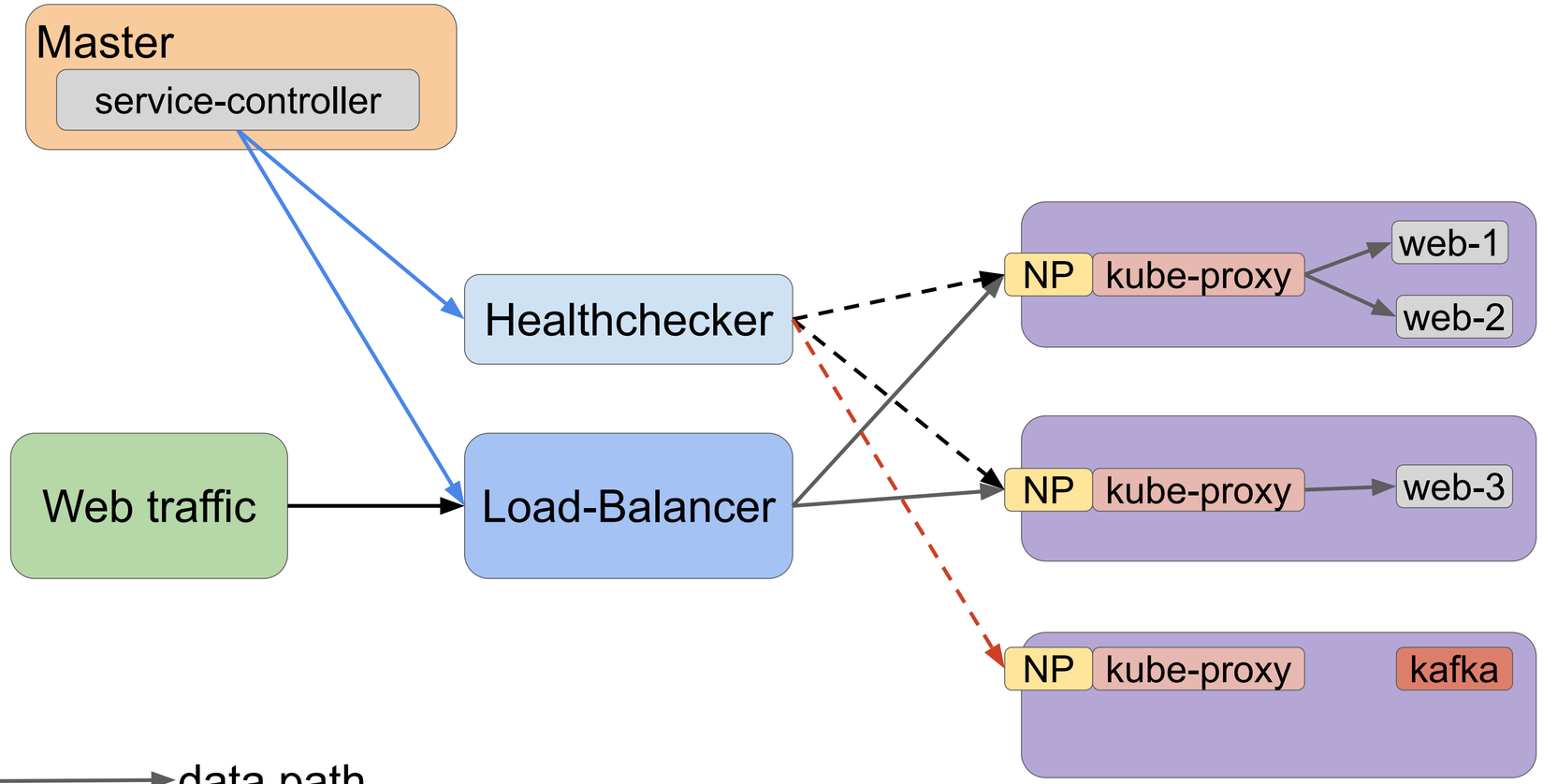
Ibernail

Inefficient Datapath & cross-application impacts

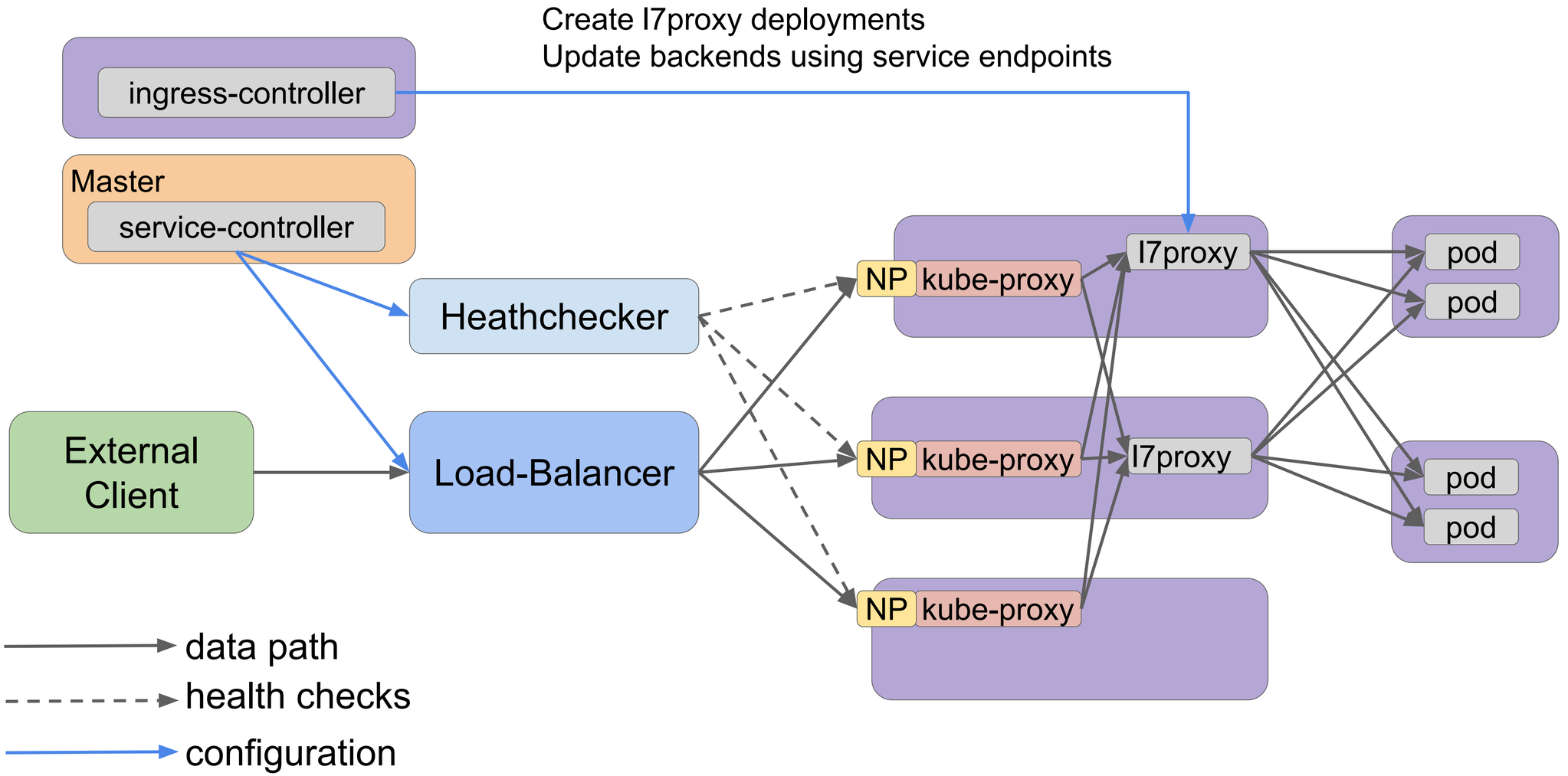


lbernail

ExternalTrafficPolicy: Local?



L7-proxy ingress controller



lbernail

Challenges

Limits

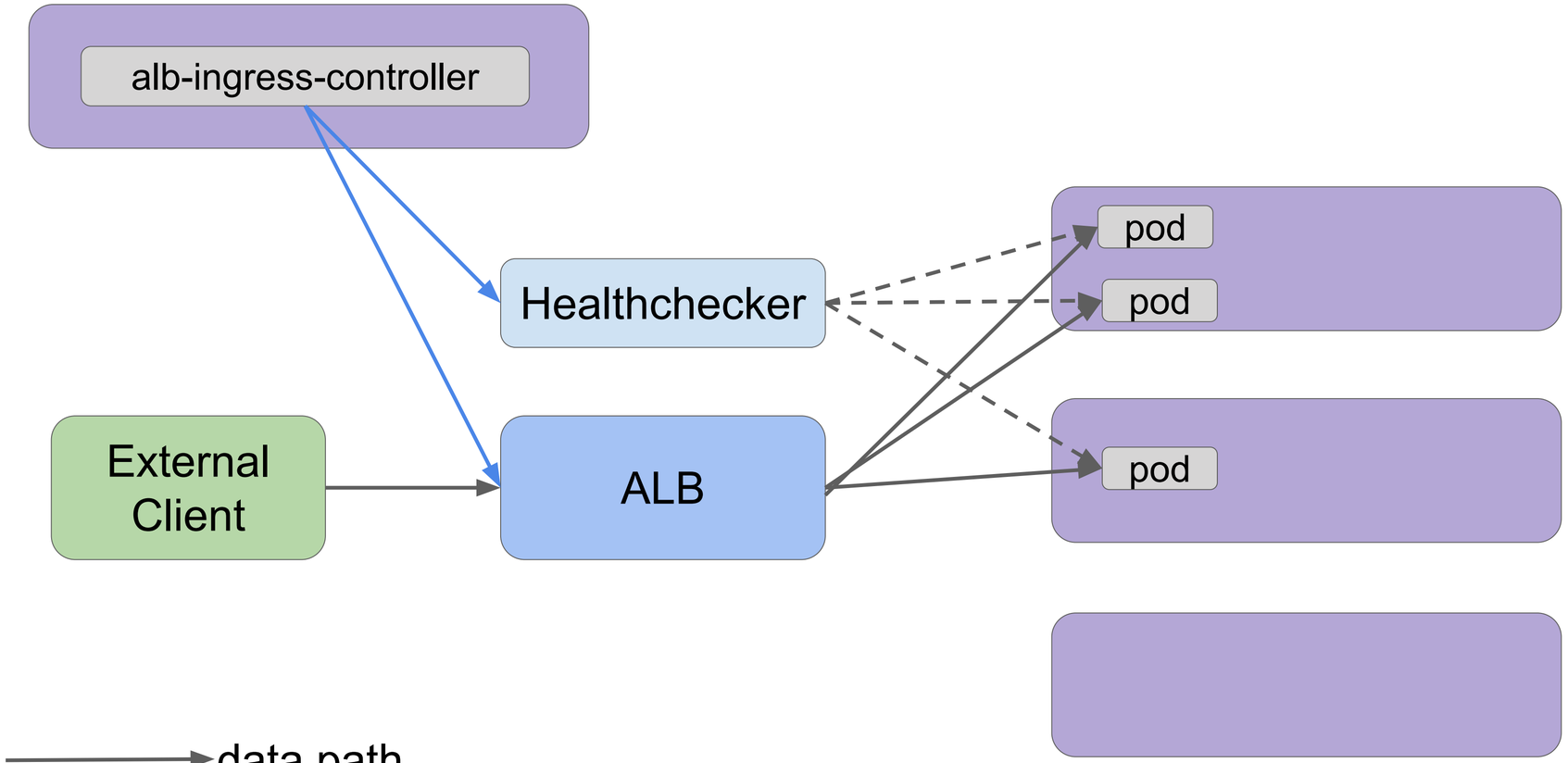
All nodes as backends (1000+)
Inefficient datapath
Cross-application impacts

Alternatives?

ExternalTrafficPolicy: Local?
> Number of nodes remains the same
> Issues with some CNI plugins

K8s ingress
> Still load-balancer based
> Need to scale ingress pods
> Still inefficient datapath

Our target: native routing



Remaining challenges

Limited to HTTP ingresses

No support for TCP/UDP

Ingress v2 should address this

Registration delay

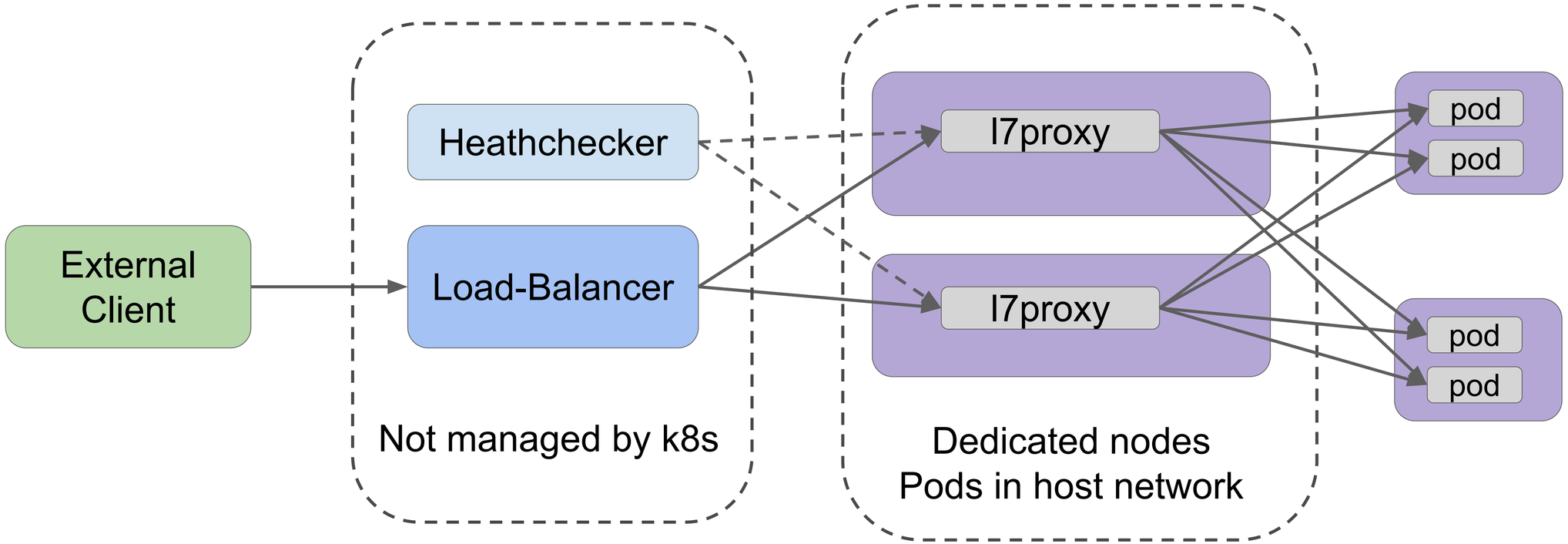
Slow registration with LB
Pod rolling-updates much faster

Mitigations
- MinReadySeconds
- Pod ReadinessGates

Workaround

TCP / Registration delay not manageable

> Dedicated gateways



Take-away

- Ingress solutions are not great at scale yet
- May require workarounds
- Definitely a very important topic for us
- The community is working on v2 Ingresses

Conclusion

A lot of other topics

- Accessing services (kube-proxy)
- DNS (it's always DNS!)
- Challenges with Stateful applications
- How to DDOS <insert ~anything> with Daemonsets
- Node Lifecycle / Cluster Lifecycle
- Deploying applications
- ...

Getting started?

“Deep Dive into Kubernetes Internals for Builders and Operators”
Jérôme Petazzoni, Lisa 2019
https://lisa-2019-10.container.training/talk.yml.html
Minimal cluster, showing interactions between main components

“Kubernetes the Hard Way”
Kelsey Hightower
https://github.com/kelseyhightower/kubernetes-the-hard-way
HA control plane with encryption

You like horror stories?

“Kubernetes the very hard way at Datadog”
https://www.youtube.com/watch?v=2dsCwp_j0yQ

“10 ways to shoot yourself in the foot with Kubernetes”
https://www.youtube.com/watch?v=QKI-JRs2RIE

“Kubernetes Failure Stories”
https://k8s.af

Key lessons

Self-managed Kubernetes is hard

> If you can, use a managed service

Networking is not easy (especially at scale)

The main challenge is not technical

> Build a team

> Transforming practices and training users is very important

Thank you

We’re hiring!
https://www.datadoghq.com/careers/

laurent@datadoghq.com
@lbernail