# FAILURE STORIES FROM THE ON-PREMISE BARE-METAL WORLD

Stephan Fudeus, Dr. David Meder-Marouelli, 1&1 Mail & Media Development & Technology GmbH

## Stephan Fudeus

- Expert for Container Platforms
- Product owner K8s platform
- 17 years of distributed systems
- Since 2017 with 1&1 Mail & Media

## Dr. David Meder-Marouelli

- Expert for Continuous Delivery
- Product owner delivery platform
- 16 years of IT experience
- Since 2015 with 1&1 Mail & Media

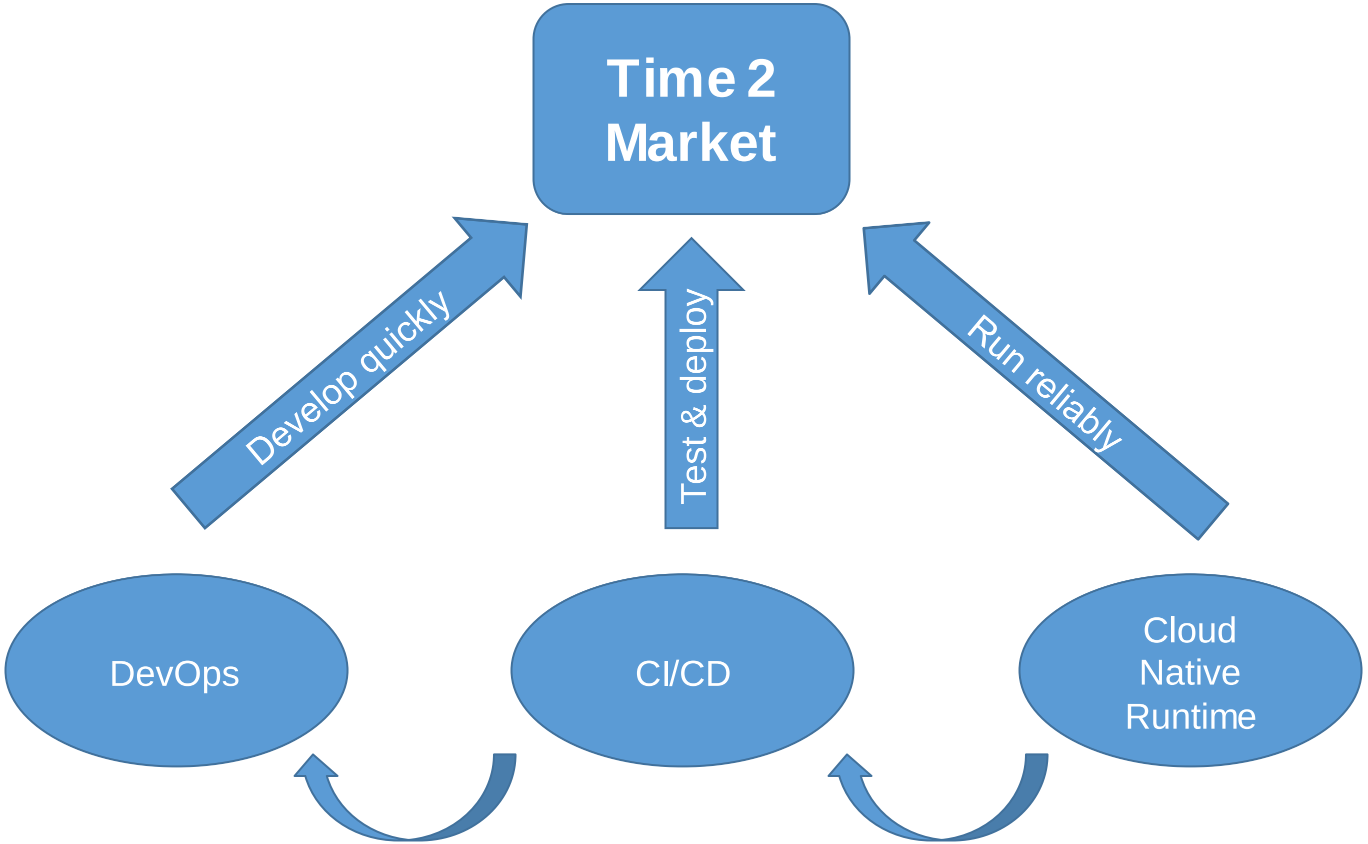
## 1&1 Mail & Media

### 1&1 Mail & Media

- Several free basic services and professional fee-based e-mail solutions
- One of the most powerful online marketing platforms
- Around 40 million active users

- CNCF End User Community Supporter

## What we do and why we do it



## Kubernetes Cluster context

- Kubernetes as centrally provided orchestration platform
  - Focus on soft multi-tenancy
    - Friendly users, but with security in mind
  - Focus on (ideally stateless) microservices
  - Fast deployment cycles
    - Weekly re-deployments

- Multiple clusters decoupled on network dimensions
  - fe/be/infrastructure, data center, live/non-live
  - bare-metal on-premise
  - non-routable podCIDR and serviceCIDR (RFC 6598 / CGNAT / 100.64.0.0/10)

## Open Source / Cloud Native Stack

HELM

PROJECT CALICO

Open Policy Agent

HashiCorp Vault

træfik

Prometheus

kubernetes

CoreDNS

f5

etcd

fluentd

GitLab

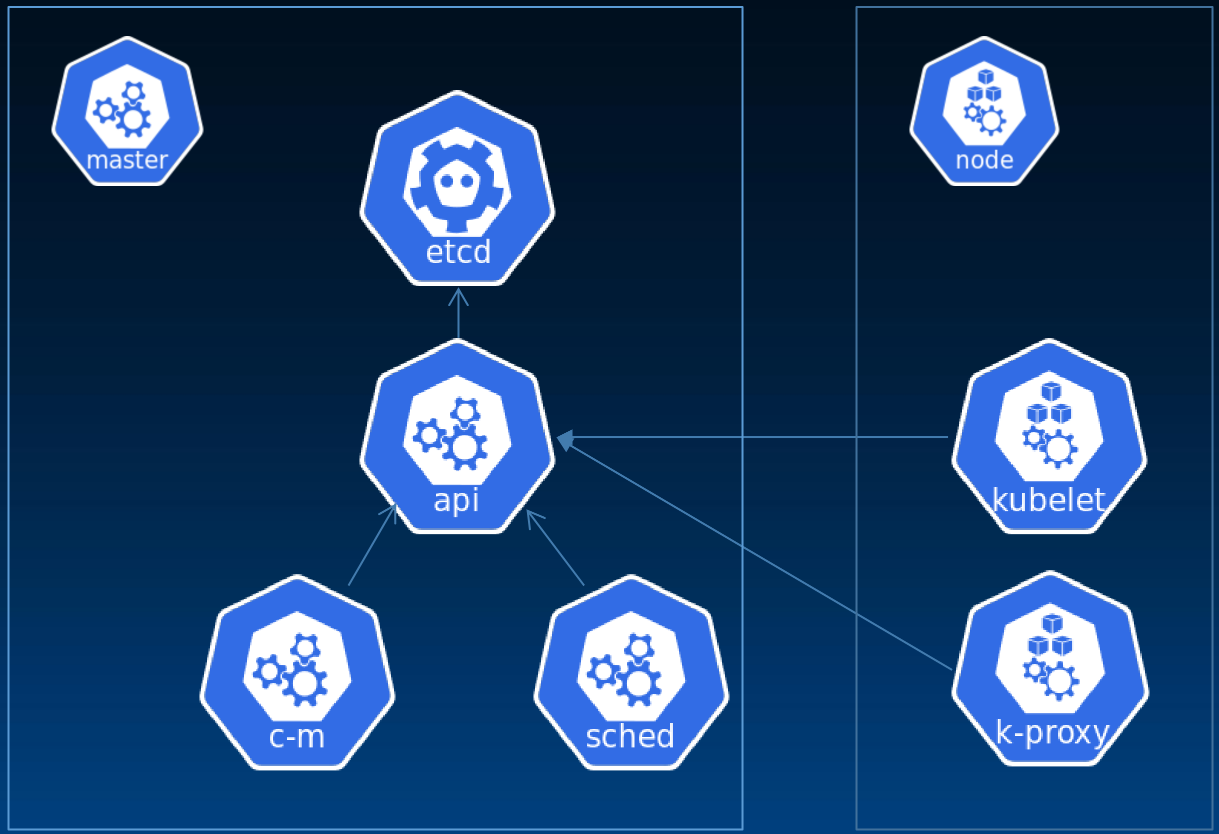
JAEGER

containerd

MetalLB

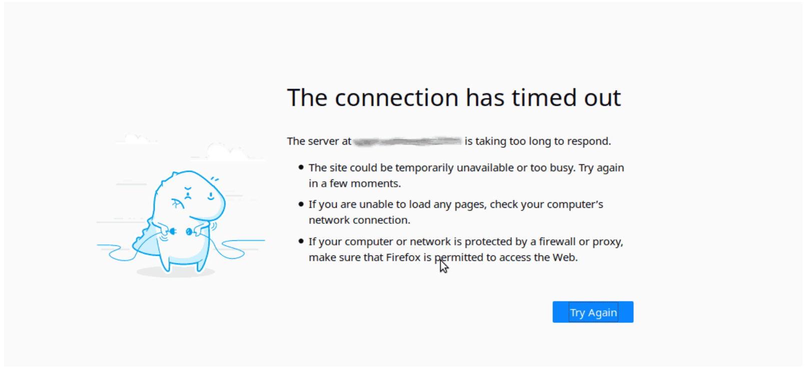
Istio

Control Plane Issues



## Why do my connections time-out?

- During periodic rolling redeploy of cluster nodes, a fraction of the new connections tend to time out during connecting.

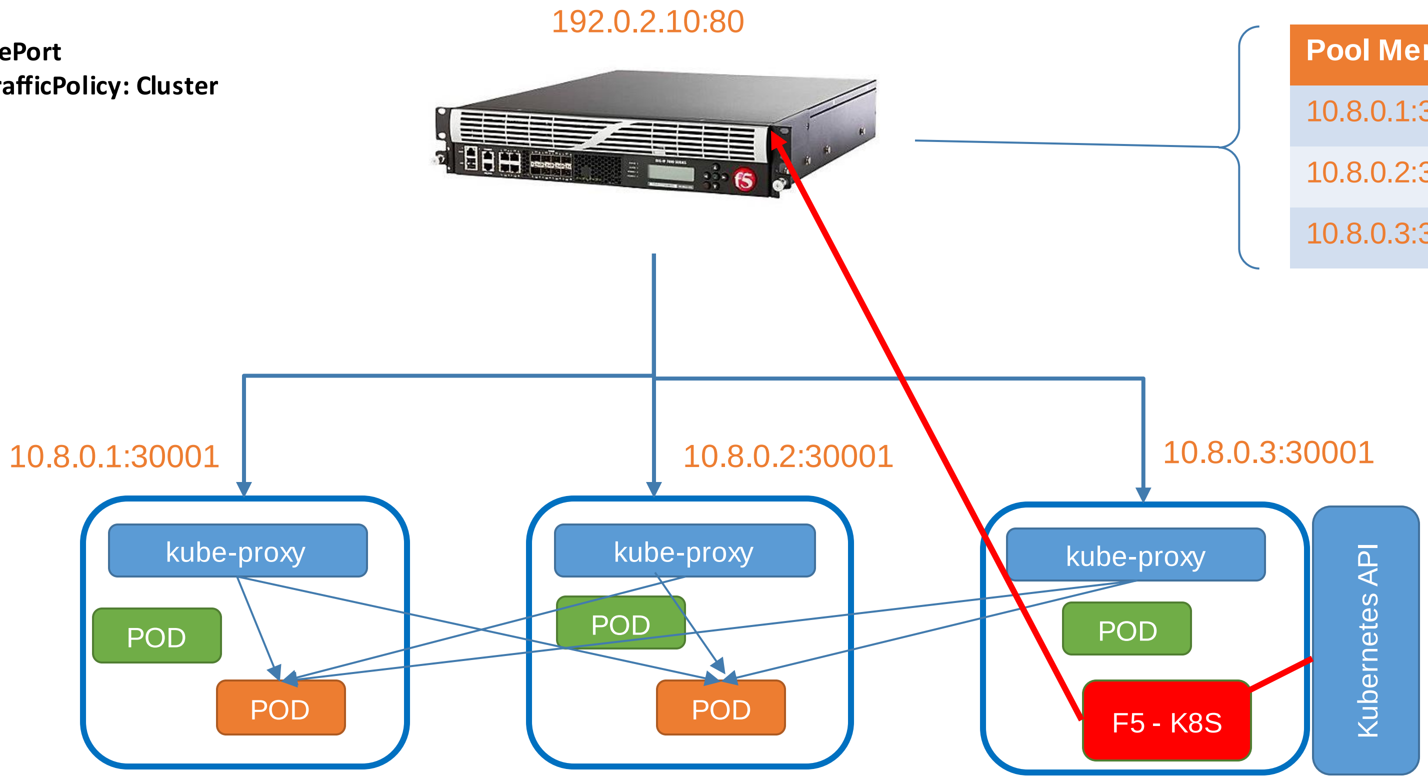

The connection has timed out

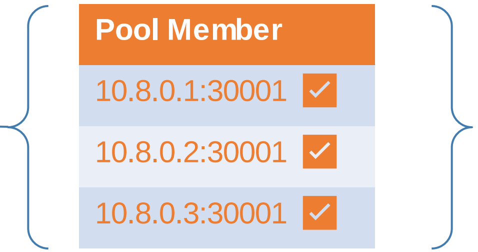
The server at [redacted] is taking too long to respond.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

# Hardware-LB as Gateway for Public Traffic

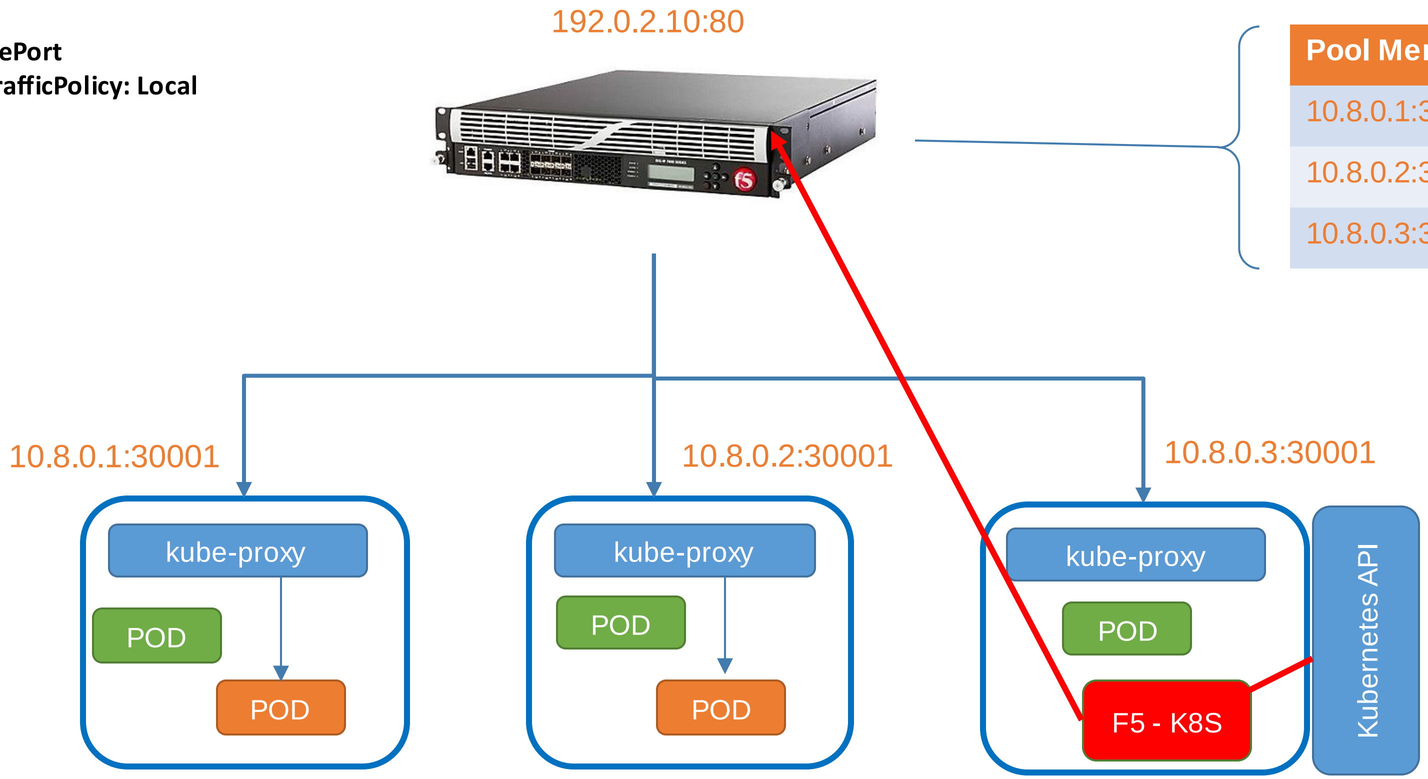
```
spec:
  type: NodePort
  externalTrafficPolicy: Cluster
```





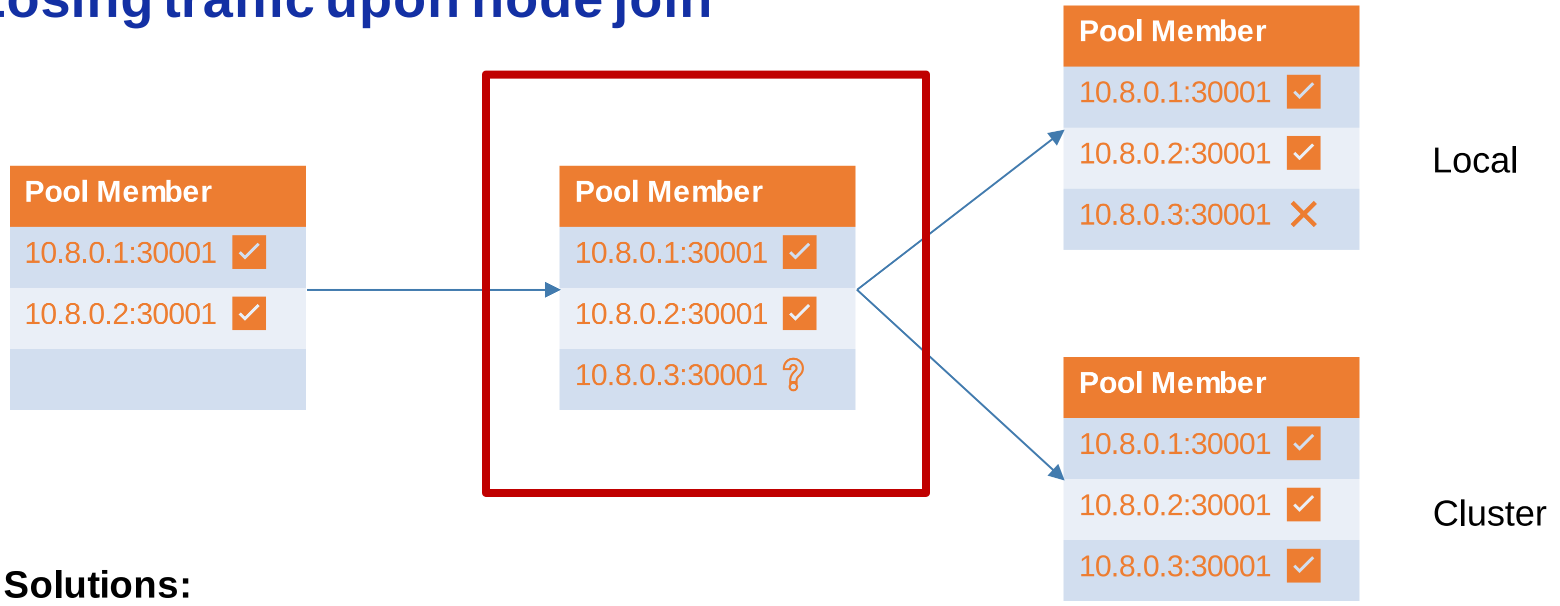
| Pool Member | |
|---|---|
| 10.8.0.1:30001 | ✓ |
| 10.8.0.2:30001 | ✓ |
| 10.8.0.3:30001 | ✓ |

```
spec:
  type: NodePort
  externalTrafficPolicy: Local
```



| Pool Member | |
|---|---|
| 10.8.0.1:30001 | ✓ |
| 10.8.0.2:30001 | ✓ |
| 10.8.0.3:30001 | ✗ |

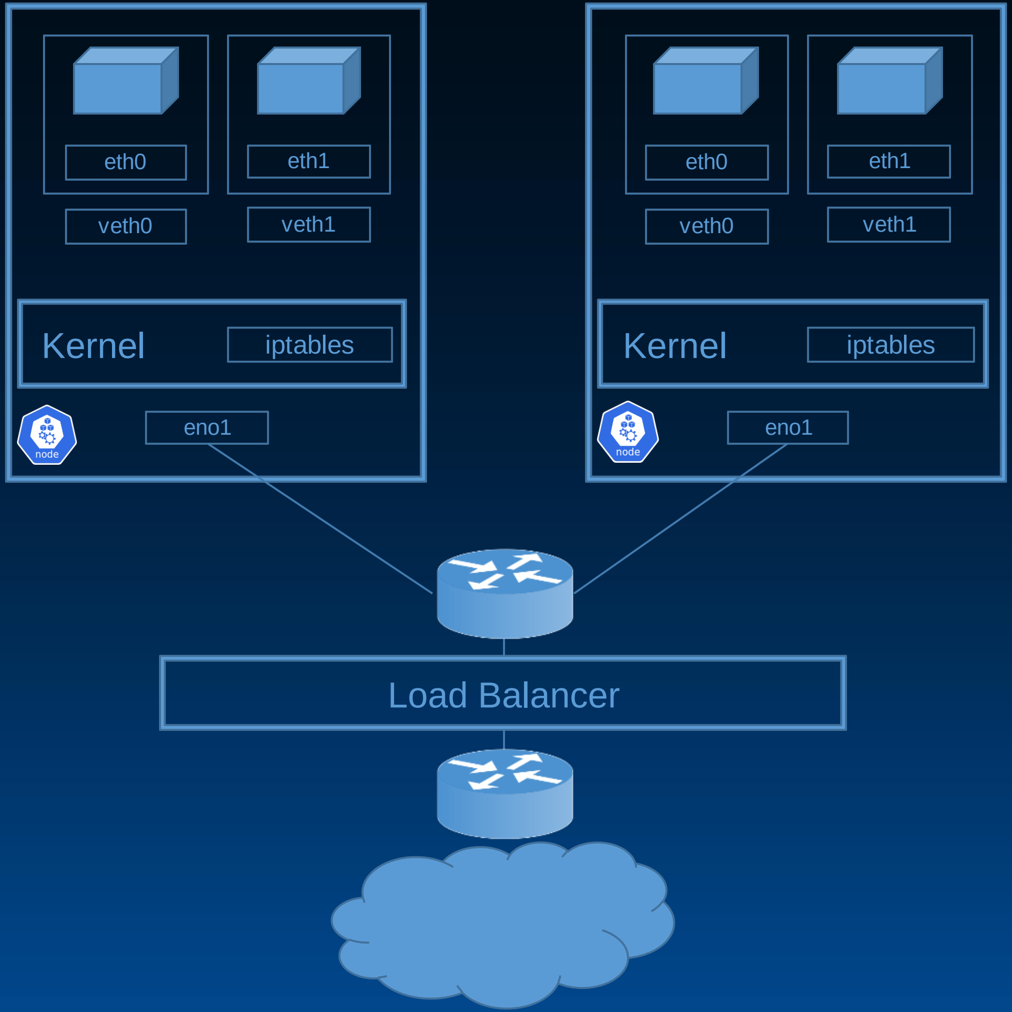
## Losing traffic upon node join



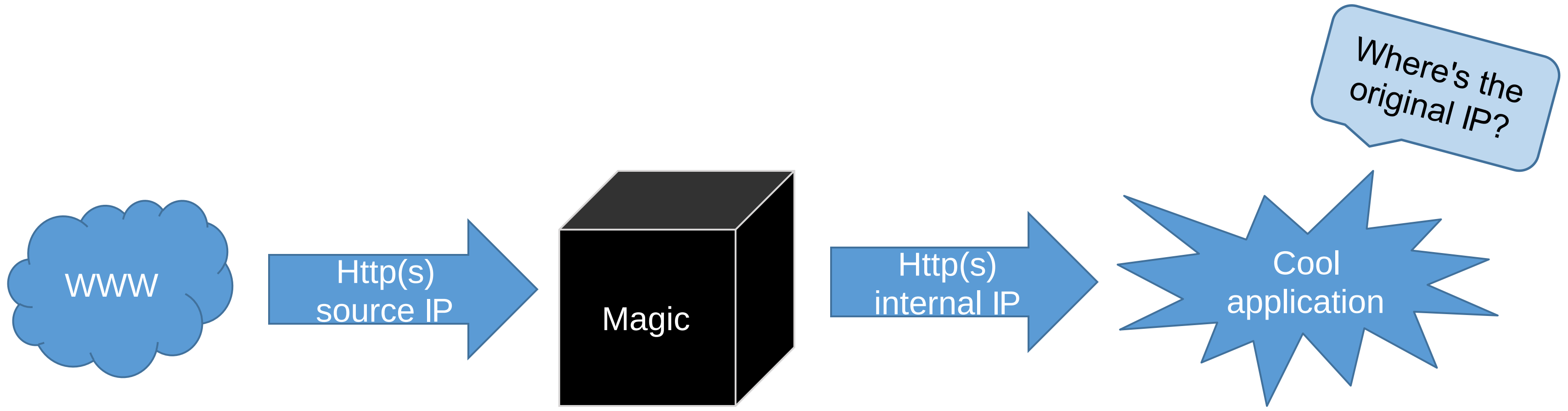
**Solutions:**

- Nodes with unknown state should not receive traffic (F5)
- Tune your ramp-up times (slow ramp)
- Tune your health checks (fast shutdown)
- Explicitly activate node after health state established (patched controller) (https://github.com/f5devcentral/f5-cccl/issues/247)
- Selection of small subset of ingress nodes
- Direct VXLAN integration with F5 → Pods as pool members, not nodes

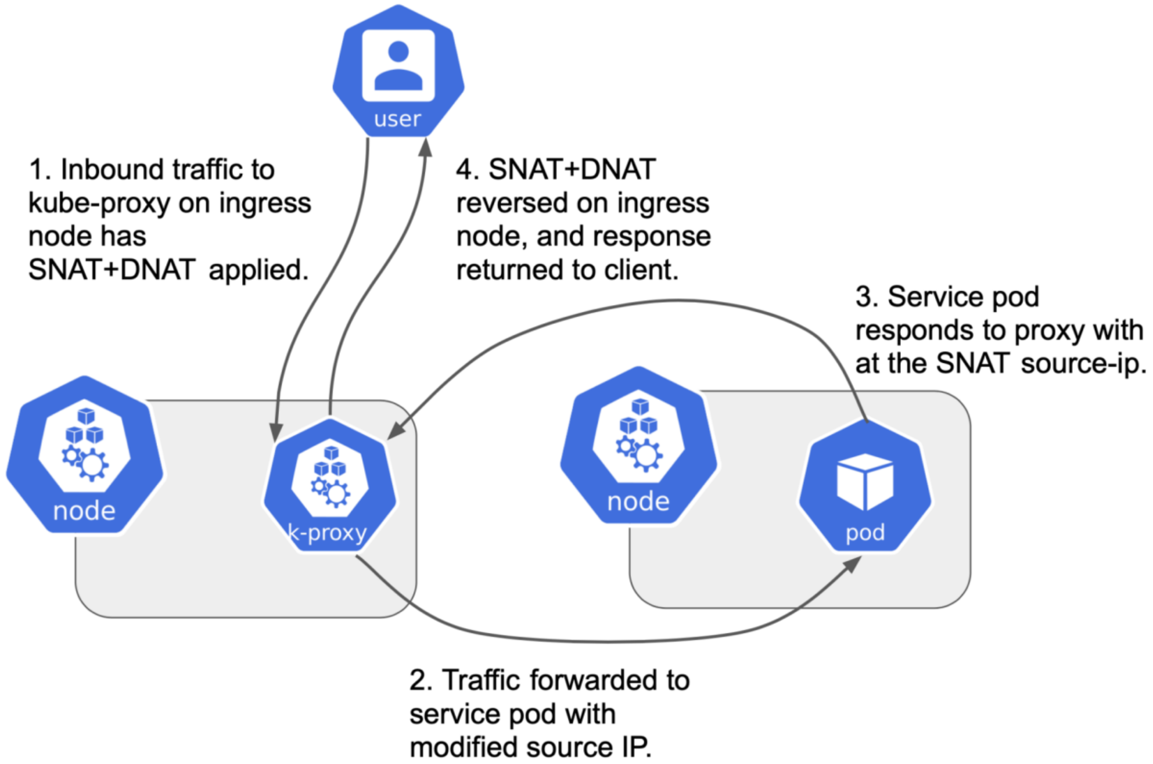
# Data Plane Issues



## Where's this source IP?

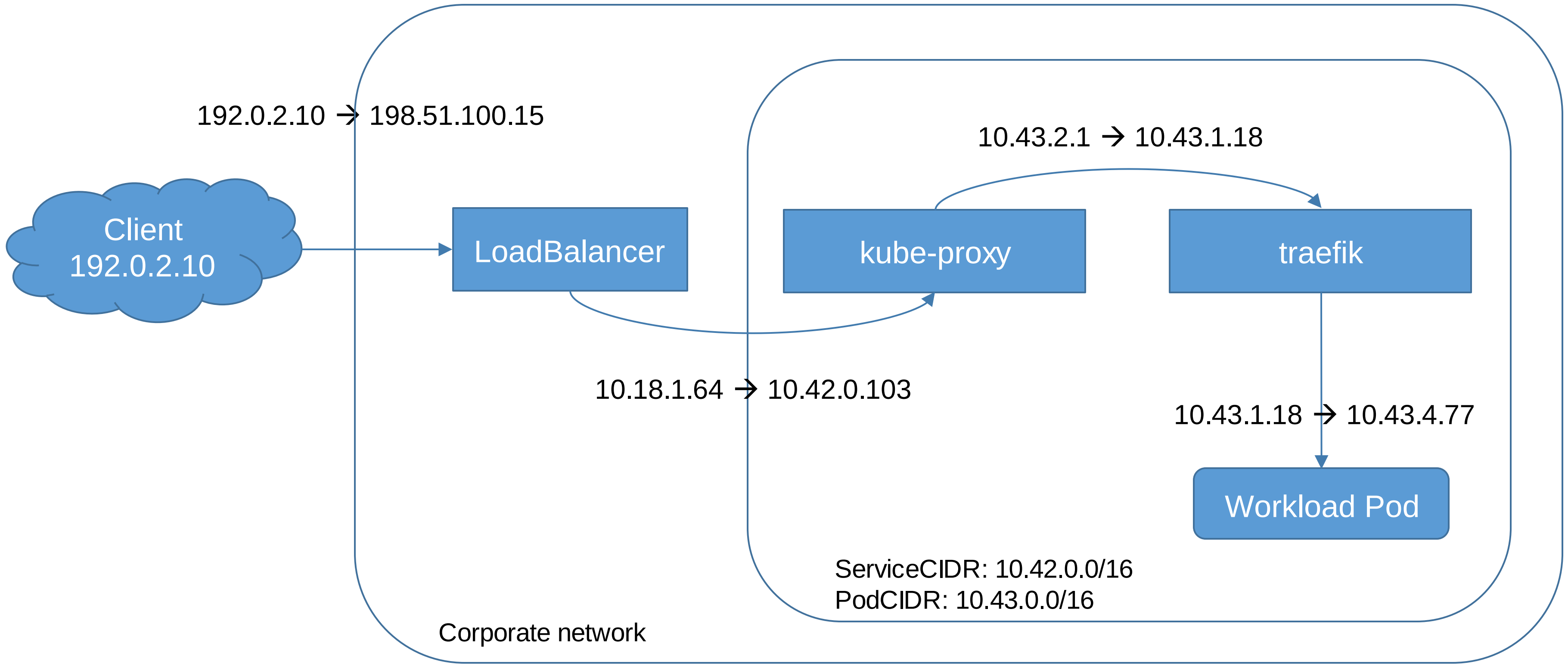


## Overlay network and kube-proxy apply NAT


user
1. Inbound traffic to kube-proxy on ingress node has SNAT+DNAT applied.
4. SNAT+DNAT reversed on ingress node, and response returned to client.
3. Service pod responds to proxy with at the SNAT source-ip.
node
k-proxy
node
pod
2. Traffic forwarded to service pod with modified source IP.

https://www.projectcalico.org/hands-on-with-calicos-ebpf-service-handling/

## Even more NAT / proxying involved



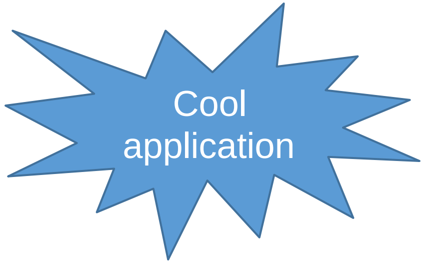
## Solutions for client sourceIP

- For HTTP(S)/GRPC traffic:
  - Forwarded/X-Forwarded-For (with optional reset on the reverse proxy / gateway)

- For other traffic:
  - externalTrafficPolicy: Local
  - different service routing (e.g. Calico eBPF)

  - But: only without forced NAT on gateway, i.e. internal traffic

# This one Pod keeps dying



```
$ kubectl get pod cool-app-74548b879c-cgp84
NAME                         READY   STATUS             RESTARTS   AGE
cool-app-74548b879c-cgp84    0/1     CrashLoopBackOff   7          14m
```

```
$ kubectl describe pod cool-app-74548b879c-cgp84
Events:
Warning  Unhealthy 6m12s (x21 over 10m)   kubelet
Liveness probe failed:
  Get http://100.65.42.41:8080/liveness:
  dial tcp 100.65.42.41:8080: connect: connection refused
```

# Calico podIP / tunnel IP collision

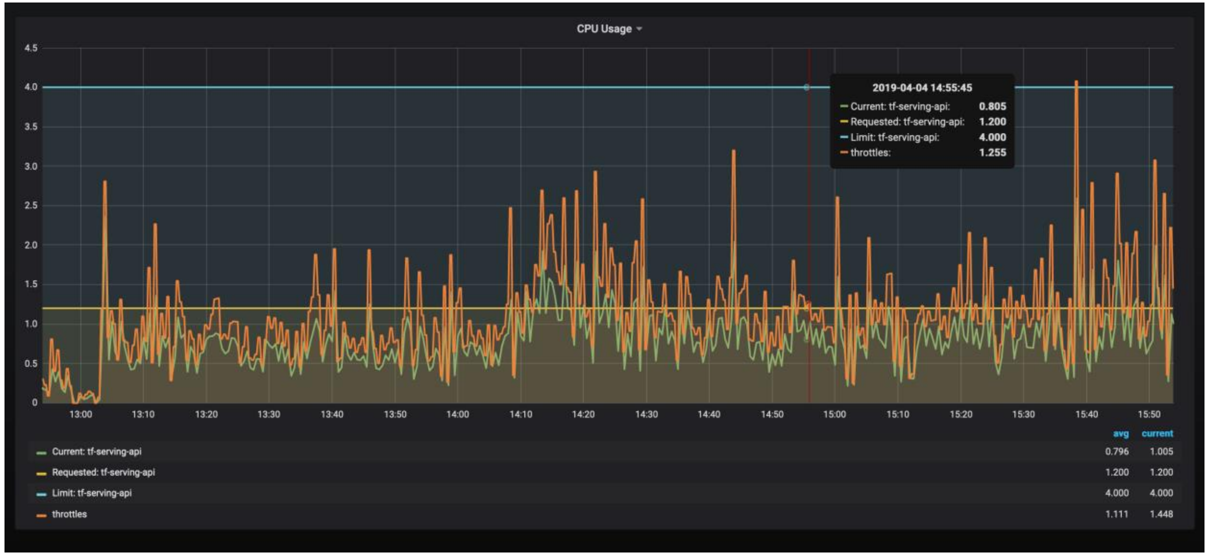
- Calico with IPIP-Encapsulation
- „small“ IP-Blocks (default /26) with > 64 pods

- IP address is used for both IPIP tunnel device AND for a pod
- Race condition managing calico ipamblocks and ipamhandles
  https://github.com/projectcalico/calico/issues/3589

- Affects reachability of the pod on the network level – showed only up in monitoring due to network based liveness check.

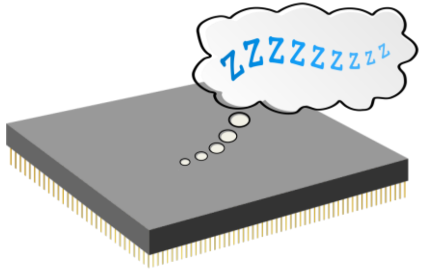
# But I get less CPU power

…than expected and requested



```
2019-04-04 14:55:45
— Current: tf-serving-api:      0.805
— Requested: tf-serving-api:    1.200
— Limit: tf-serving-api:        4.000
— throttles:                    1.255
```

```
                                 avg    current
— Current: tf-serving-api       0.796   1.005
— Requested: tf-serving-api     1.200   1.200
— Limit: tf-serving-api         4.000   4.000
— throttles                     1.111   1.448
```



CPU Limit

Actual CPU

https://user-images.githubusercontent.com/29459870/55629839-3353b680-5782-11e9-8dff-d57a2ae937cb.png
https://github.com/kubernetes/kubernetes/issues/67577

## … because of a kernel bug

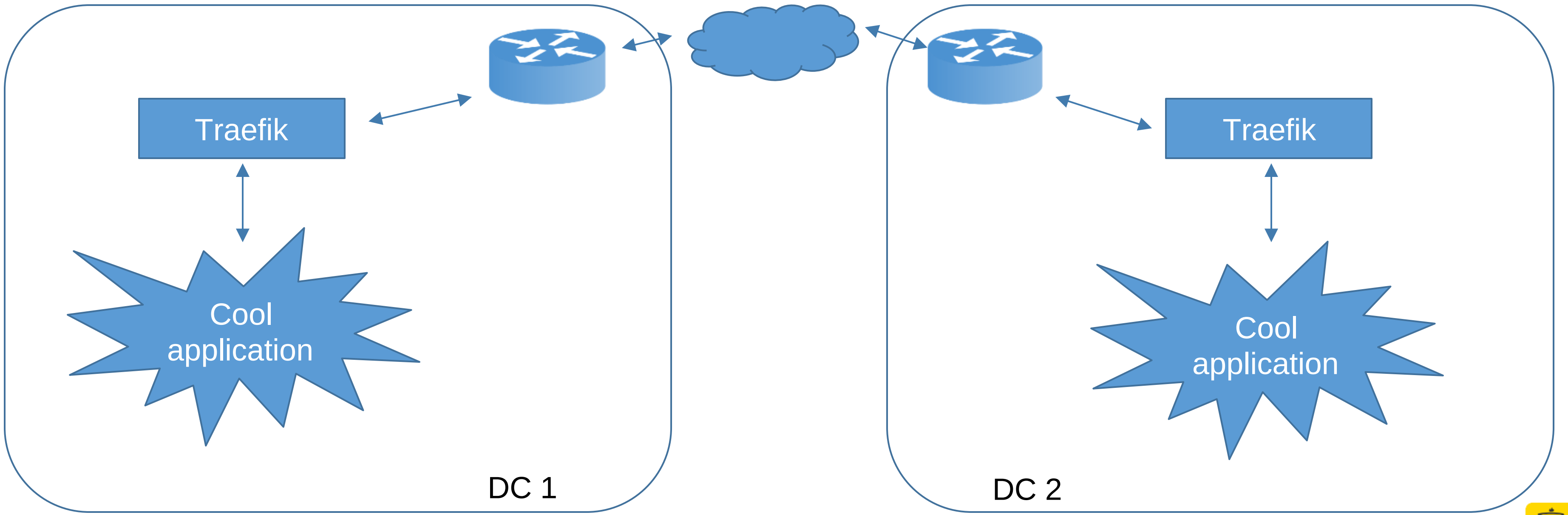
- Linux kernel CFS bandwidth control manages pod cpu limits
- Requests and limits are controlled by different mechanisms
- Multi-threaded, non cpu-bound workloads
- Several users even disabled limits support

- Multiple kernel patches between mid 2018 and September 2019
  … landed in e.g. ContainerLinux in November 2019

- Links
  - https://github.com/kubernetes/kubernetes/issues/67577
  - https://github.com/coreos/bugs/issues/2623
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=512ac999
  - https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=de53fd7ae

## And I lose active tcp connections …

- … when you do cluster-redeployments
- … when my load changes, but only in short spikes



## … because

- Service-IPs are announced via BGP and use ECMP
  - Routing tables change whenever the announcements change and cause resets of TCP connections
  - **Solution**: Reduce speaker to stabilize changes
- Autoscaling traefik with `externalTrafficPolicy: Local`
  - Scaling up/down traefik while having less replica than speakers cause ECMP changes
  - **Solution**: traefik.hpa.minReplica > speaker.replicas
  - Additional caveat: Check balancing of traefik replicas across speakers

## Conclusion

- Remaining open issues
  - Recent issue with network ingress
  - Affinities not always act as expected
  - Kube-proxy overload through single crashing 20-pod app

- Architectural implications
  - Design for failure
  - Have metrics for post mortems
  - Issues are lurking in both infrastructure and applications
  - Care about proper liveness/readiness checks

## References

- „Official“ compilation of k8s failure stories
  https://k8s.af/ by Henning Jacobs