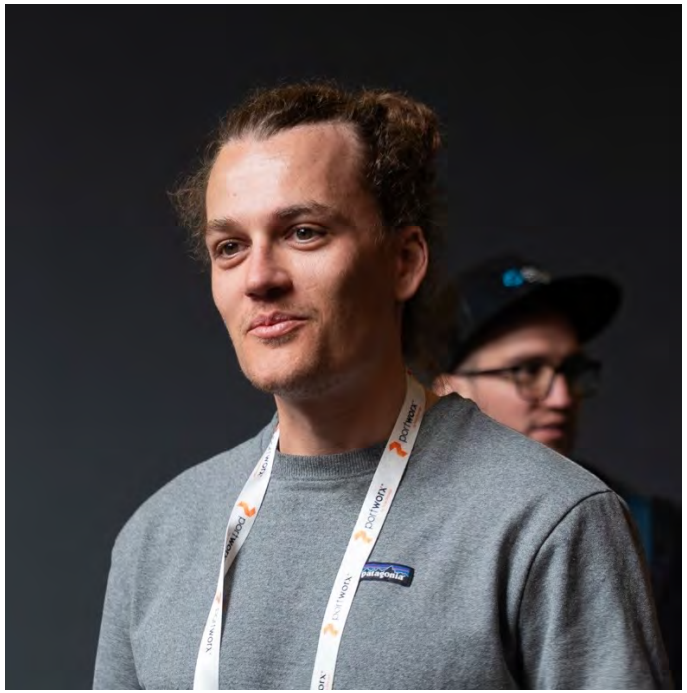
Dig Smart: Creating A Reliable Cloud-Native DNS Service

Joel Studler, Fabian Schulz & Georgios Daskalopoulos

swisscom

Joel Studler
DevOps Engineer

joel.studler@swisscom.com



Fabian Schulz
DevOps Engineer

fabian.schulz1@swisscom.com

Georgios Daskalopoulos
DevOps Engineer

gdasky@gmail.com

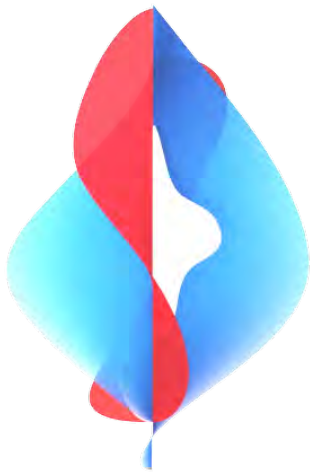
Part 1: Creating A Reliable Cloud Native DNS Service

by Fabian Schulz & Joel Studler

Part 2: Resilience Testing on Cloud Native DNS

University of Zurich[UZH]

by Georgios Daskalopoulos



swisscom

Context & Related Talks

5G - driving our journey from Telco to TechCo
by Swisscom CTIO Mark Düsener at Connect Conference 2022
https://www.youtube.com/watch?v=hND7TiXJED8

Evolving GitOps: Harnessing Kubernetes Resource Model for 5G
by Ashan Senevirathne and Joel Studler at Open Source Summit 2024
https://www.youtube.com/watch?v=35-fE_gHDjw

From Weeks to Seconds: Scaling Network Automation with Kubernetes Operators
by Alessio Diamanti and Adrian Kurt at ContainerDays 2025
https://www.youtube.com/watch?v=RytDzZYDQTU

DNS in 5G Core

Specific Private Zones

Domains used in Mobile Network only such as 3gppnetwork.org

Moderate Throughput

10s to 100s of Requests/second

Low Latency

DNS is an important factor in the overall performance of the Mobile Network

Requirements for the 5G Core DNS Service

Proximity to Consumer
Minimal amount of hops between 5G Core and DNS

✗ No SaaS allowed

Fully Automated
GitOps driven and automated provisioning of DNS records

✗ No manual interaction allowed

Geo Redundant & HA
Spread across multiple K8s clusters and geo regions to increase reliability

✗ No singletons

Support of Advanced DNS features
Resource Records such as NAPTR and SRV supported for e.g. SIP Phone Calls

✗ Need to go beyond A and CNAME

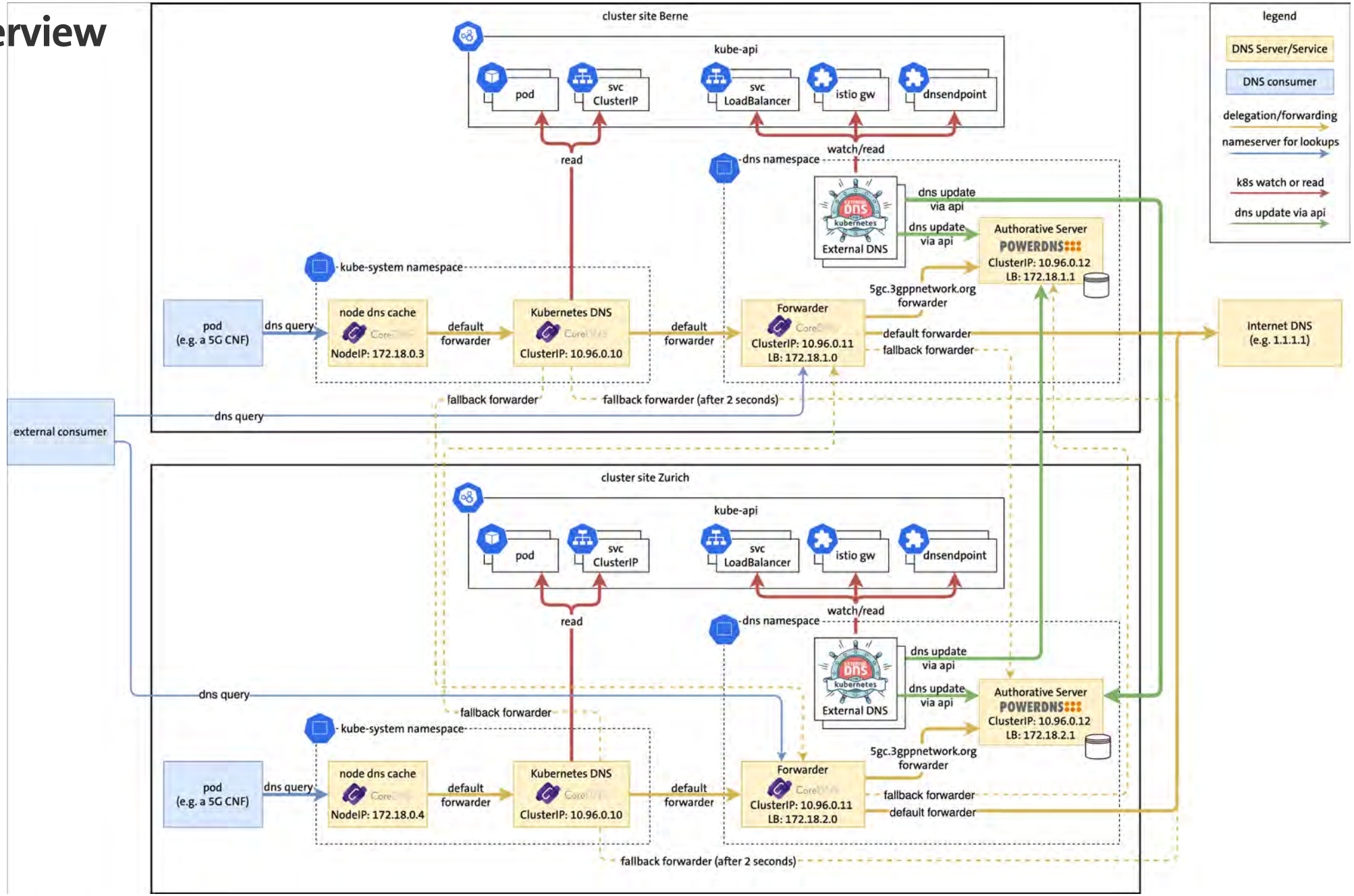
K8s integration with ExternalDNS
The System leverages Kubernetes Patterns such as CRs and Operators

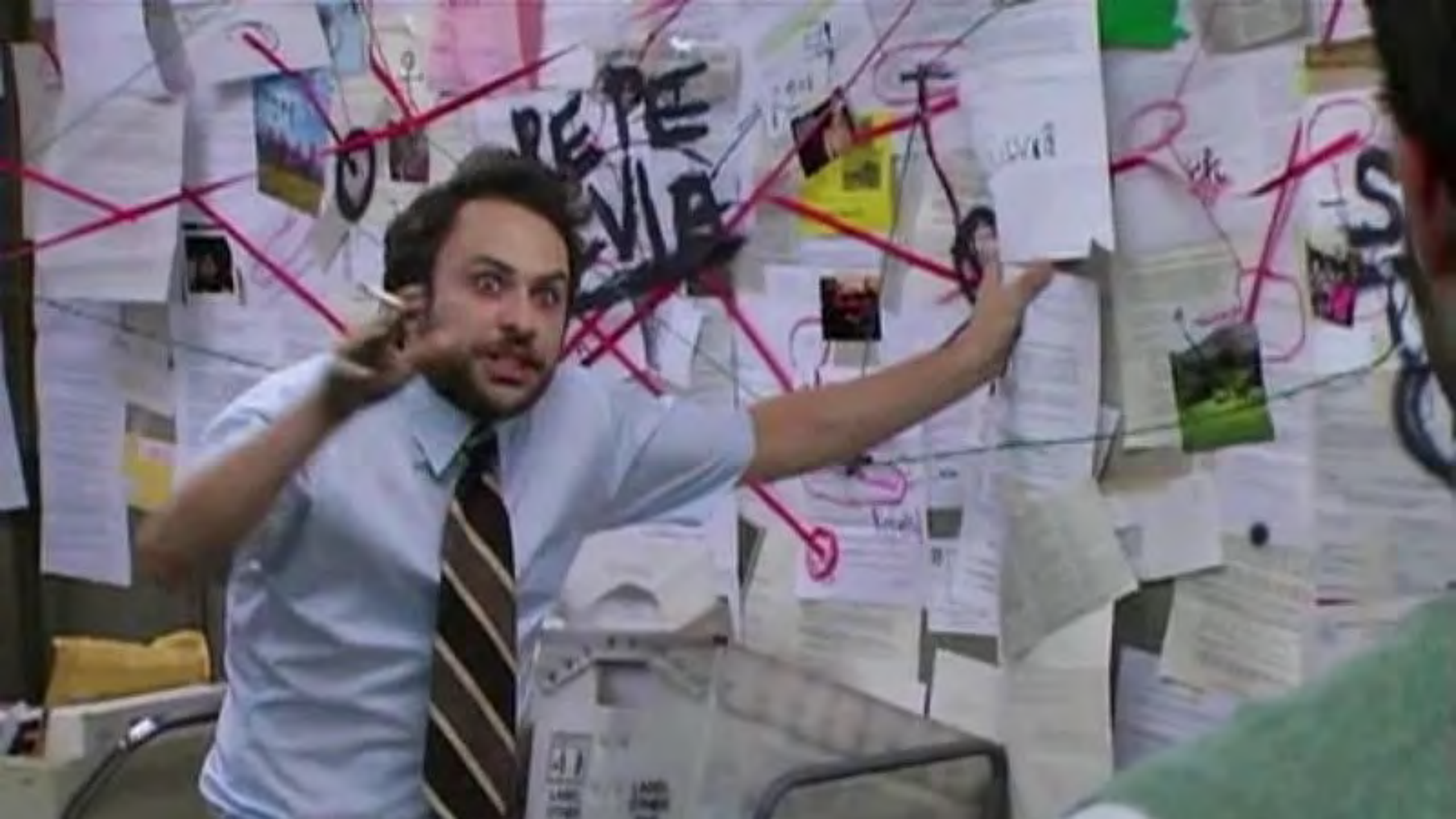
✗ No CRUD outside kube-api

Minimal Amount of SPOFs
Share nothing by removing single points of failure from the System

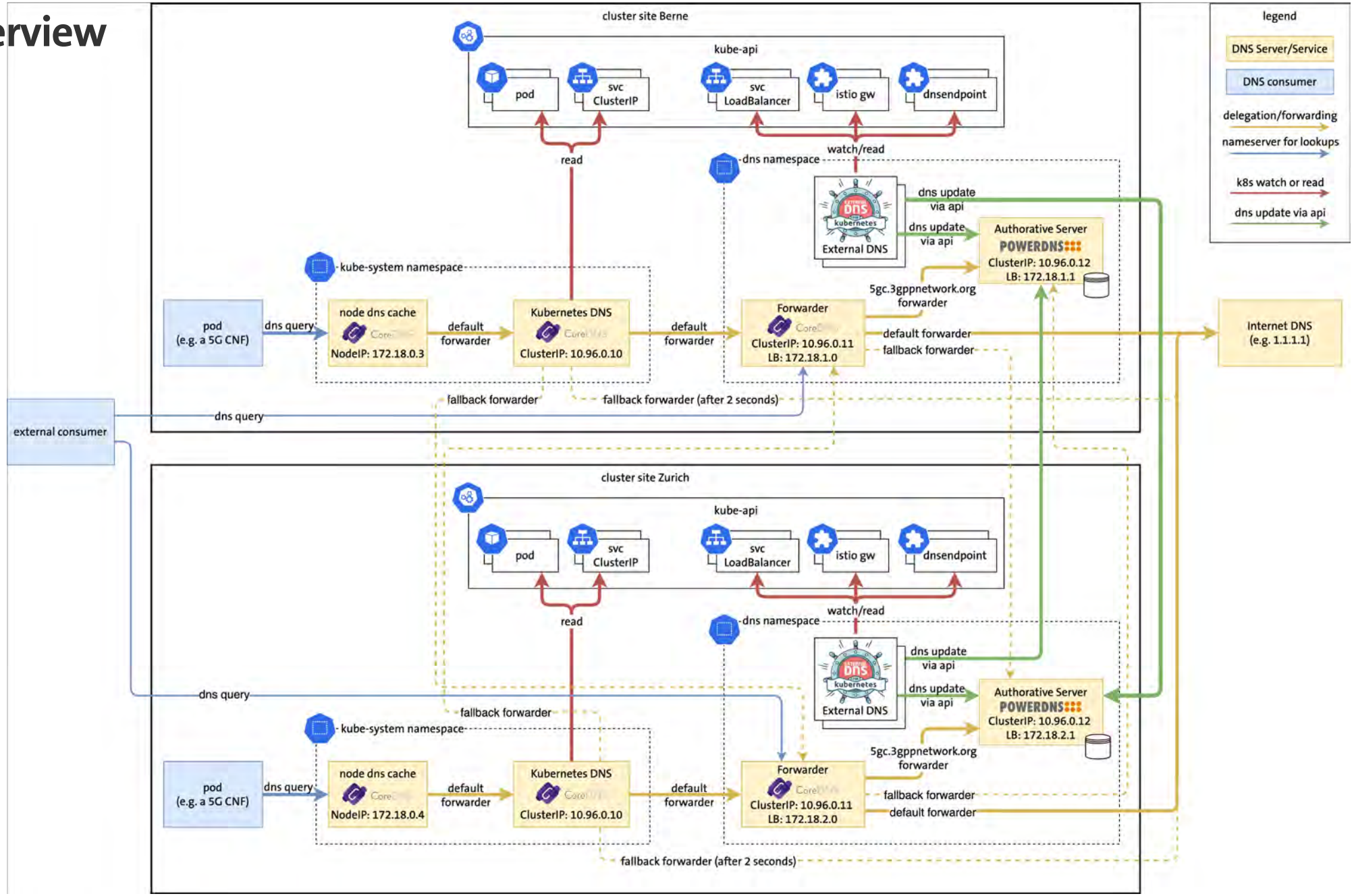
✗ No shared mgmt system

Overview

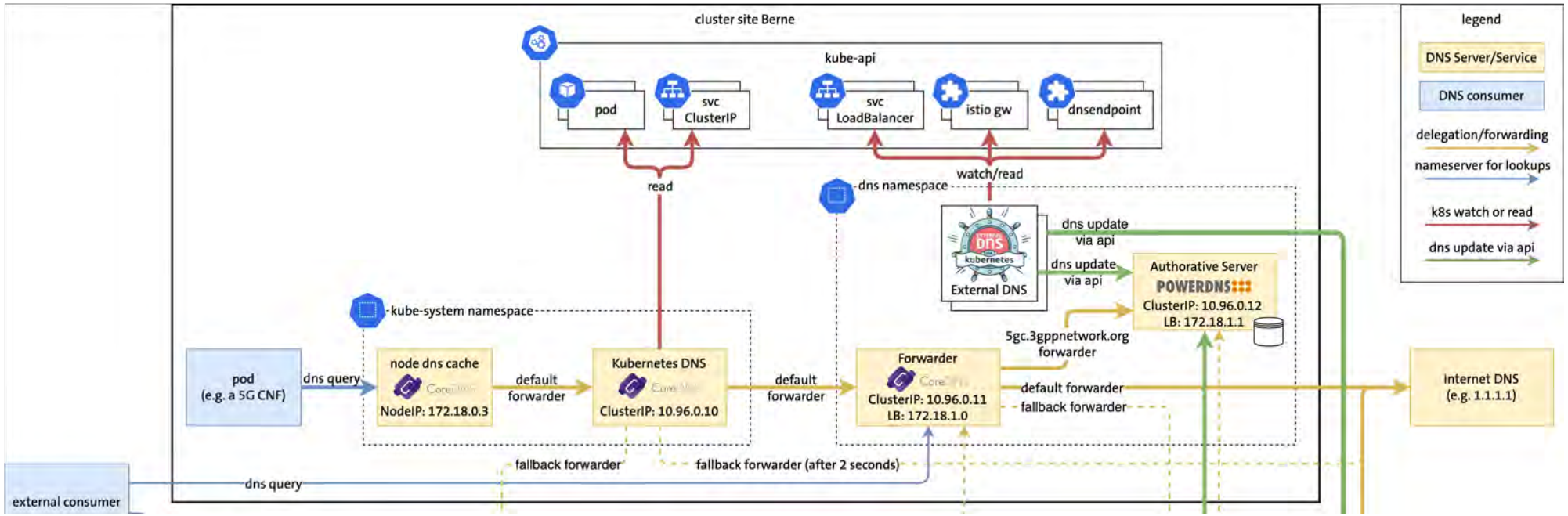

cluster site Berne
kube-api
pod
svc ClusterIP
svc LoadBalancer
istio gw
dnsendpoint
read
watch/read
dns namespace
External DNS
dns update via api
dns update via api
Authorative Server
POWERDNS
ClusterIP: 10.96.0.12
LB: 172.18.1.1
kube-system namespace
pod (e.g. a 5G CNF)
dns query
node dns cache
NodeIP: 172.18.0.3
default forwarder
Kubernetes DNS
ClusterIP: 10.96.0.10
default forwarder
Forwarder
ClusterIP: 10.96.0.11
LB: 172.18.1.0
5gc.3gppnetwork.org forwarder
default forwarder
fallback forwarder
Internet DNS (e.g. 1.1.1.1)
fallback forwarder
fallback forwarder (after 2 seconds)
dns query
external consumer
cluster site Zurich
kube-api
pod
svc ClusterIP
svc LoadBalancer
istio gw
dnsendpoint
read
watch/read
dns namespace
External DNS
dns update via api
dns update via api
Authorative Server
POWERDNS
ClusterIP: 10.96.0.12
LB: 172.18.2.1
dns query
fallback forwarder
kube-system namespace
pod (e.g. a 5G CNF)
dns query
node dns cache
NodeIP: 172.18.0.4
default forwarder
Kubernetes DNS
ClusterIP: 10.96.0.10
default forwarder
Forwarder
ClusterIP: 10.96.0.11
LB: 172.18.2.0
5gc.3gppnetwork.org forwarder
fallback forwarder
default forwarder
fallback forwarder (after 2 seconds)
legend
DNS Server/Service
DNS consumer
delegation/forwarding
nameserver for lookups
k8s watch or read
dns update via api



Overview


cluster site Berne
kube-api
pod
svc ClusterIP
svc LoadBalancer
istio gw
dnsendpoint
read
watch/read
dns namespace
External DNS
dns update via api
dns update via api
Authorative Server
POWERDNS
ClusterIP: 10.96.0.12
LB: 172.18.1.1
kube-system namespace
pod (e.g. a 5G CNF)
dns query
node dns cache
NodeIP: 172.18.0.3
default forwarder
Kubernetes DNS
ClusterIP: 10.96.0.10
default forwarder
Forwarder
ClusterIP: 10.96.0.11
LB: 172.18.1.0
5gc.3gppnetwork.org forwarder
default forwarder
fallback forwarder
Internet DNS (e.g. 1.1.1.1)
fallback forwarder
fallback forwarder (after 2 seconds)
external consumer
dns query
cluster site Zurich
kube-api
pod
svc ClusterIP
svc LoadBalancer
istio gw
dnsendpoint
read
watch/read
dns namespace
External DNS
dns update via api
dns update via api
Authorative Server
POWERDNS
ClusterIP: 10.96.0.12
LB: 172.18.2.1
dns query
fallback forwarder
kube-system namespace
pod (e.g. a 5G CNF)
dns query
node dns cache
NodeIP: 172.18.0.4
default forwarder
Kubernetes DNS
ClusterIP: 10.96.0.10
default forwarder
Forwarder
ClusterIP: 10.96.0.11
LB: 172.18.2.0
5gc.3gppnetwork.org forwarder
fallback forwarder
default forwarder
fallback forwarder (after 2 seconds)
legend
DNS Server/Service
DNS consumer
delegation/forwarding
nameserver for lookups
k8s watch or read
dns update via api

Overview Single Cluster



In-Cluster Service Discovery in Kubernetes

CoreDNS (https://coredns.io)

Reads from kube-api

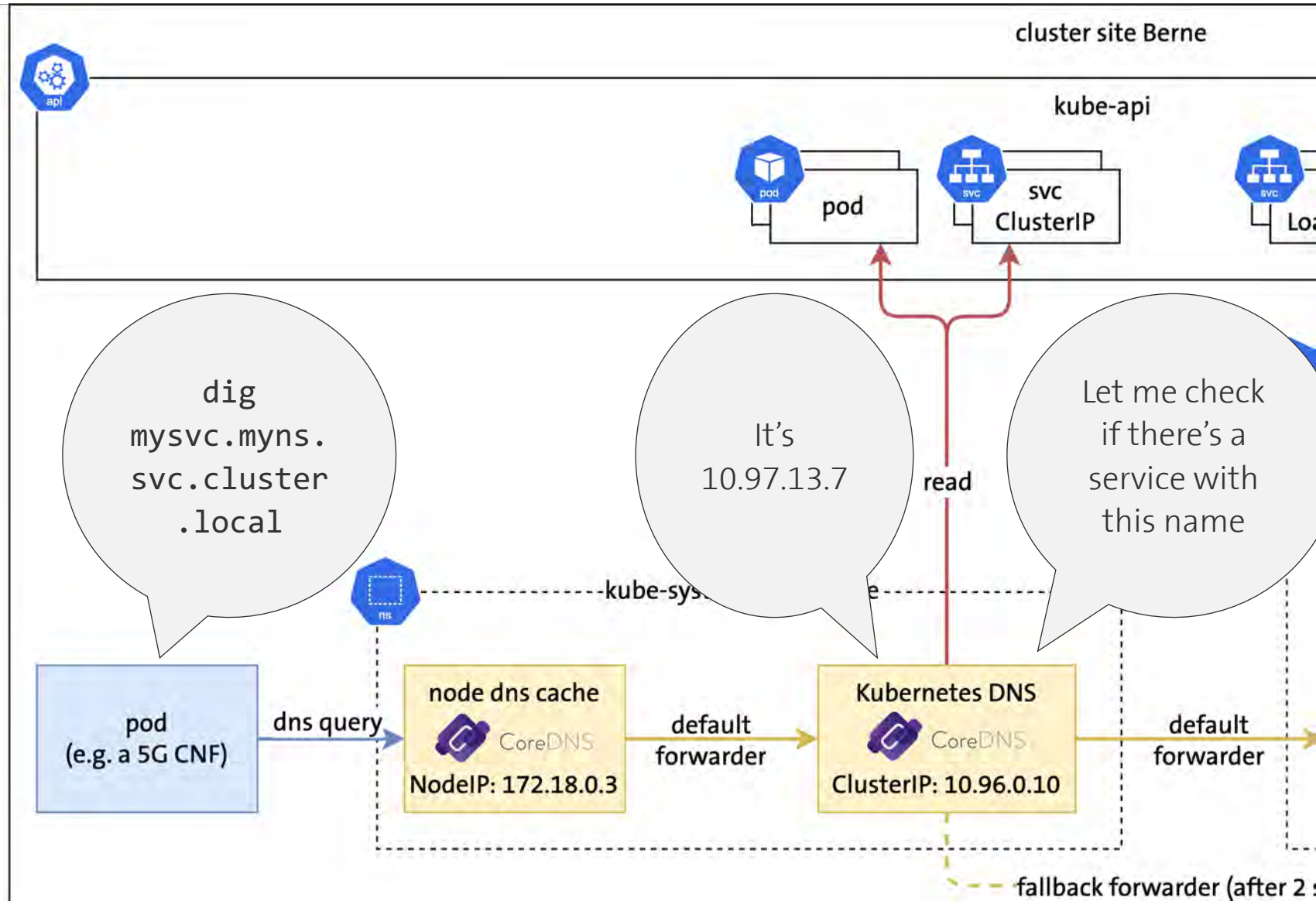
Features:

✓ In-Cluster Service Discovery

Missing:

✗ Not exposed outside of K8s

✗ No custom Resource Records



In-Cluster Service Discovery in Kubernetes: Resources

Kubernetes DNS: https://kubernetes.io/docs/concepts/services-networking/dns-pod-service

Reserved ClusterIP Address assignment: https://kubernetes.io/docs/concepts/services-networking/cluster-ip-allocation/#why-do-you-need-to-reserve-service-cluster-ips

Node Cache: https://kubernetes.io/docs/tasks/administer-cluster/nodelocaldns

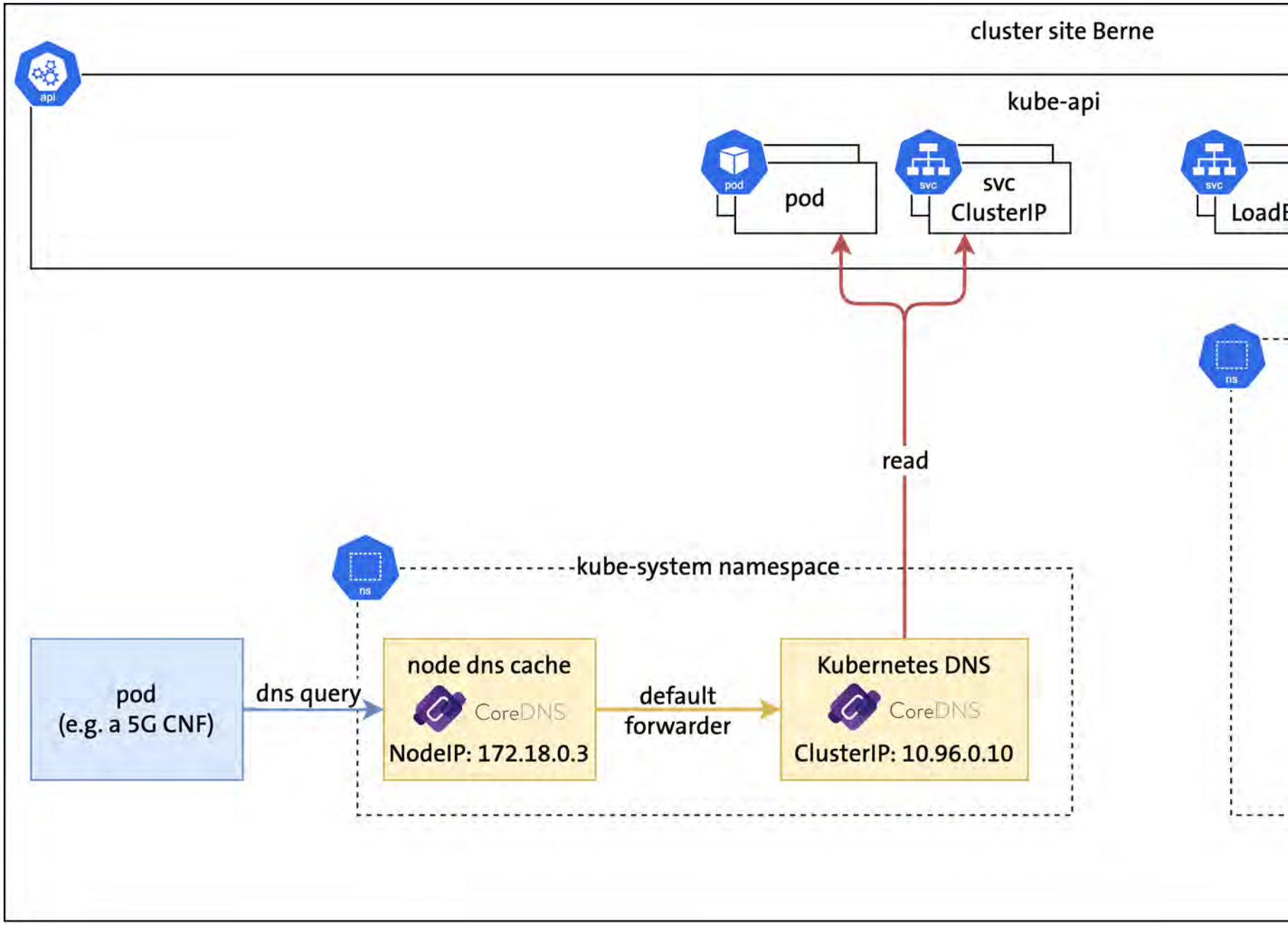
Debugging Kubernetes DNS: https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution

Customize DNS Service: https://kubernetes.io/docs/tasks/administer-cluster/dns-custom-nameservers

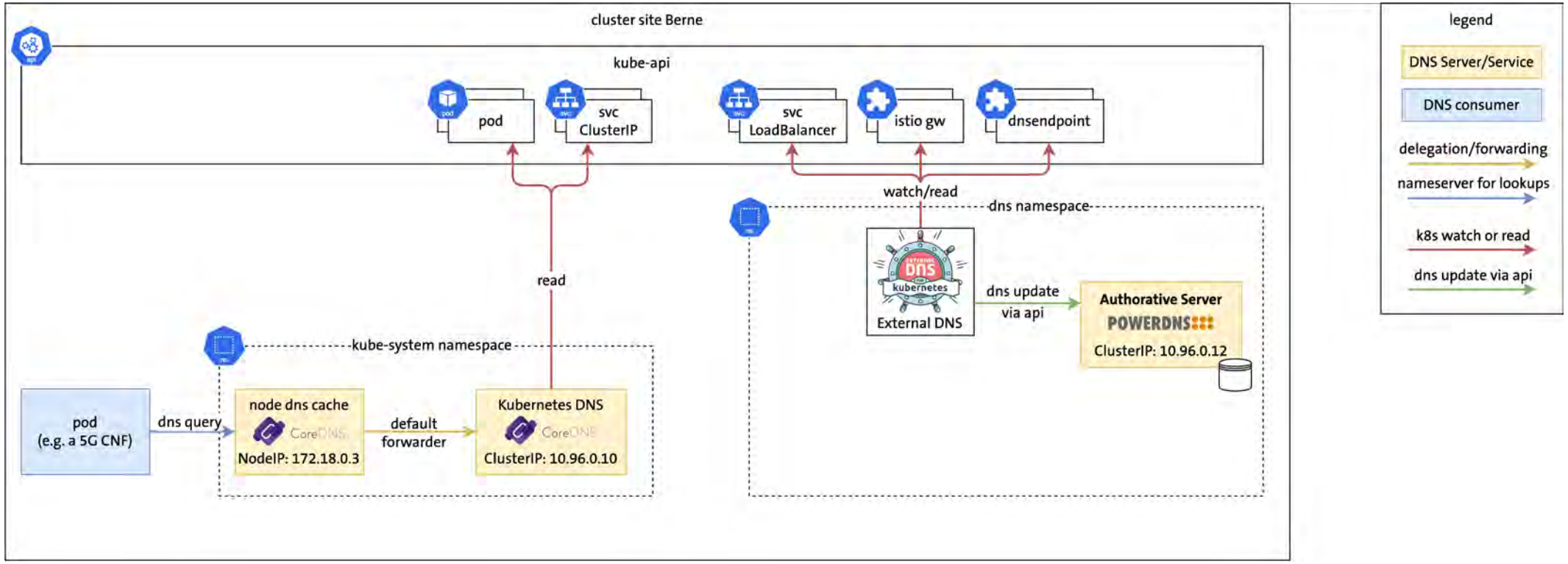
Requirements for Authoritative Server

Requirement	CoreDNS	PowerDNS Authoritative	SaaS
ExternalDNS* Support for K8s integration	✓	✓	✓
A & CNAME Resource Records	✓	✓	✓
NAPTR Resource Records (e.g. for SIP phone calls)	✗	✓**	✓**
Proximity to Consumer	✓	✓	✗

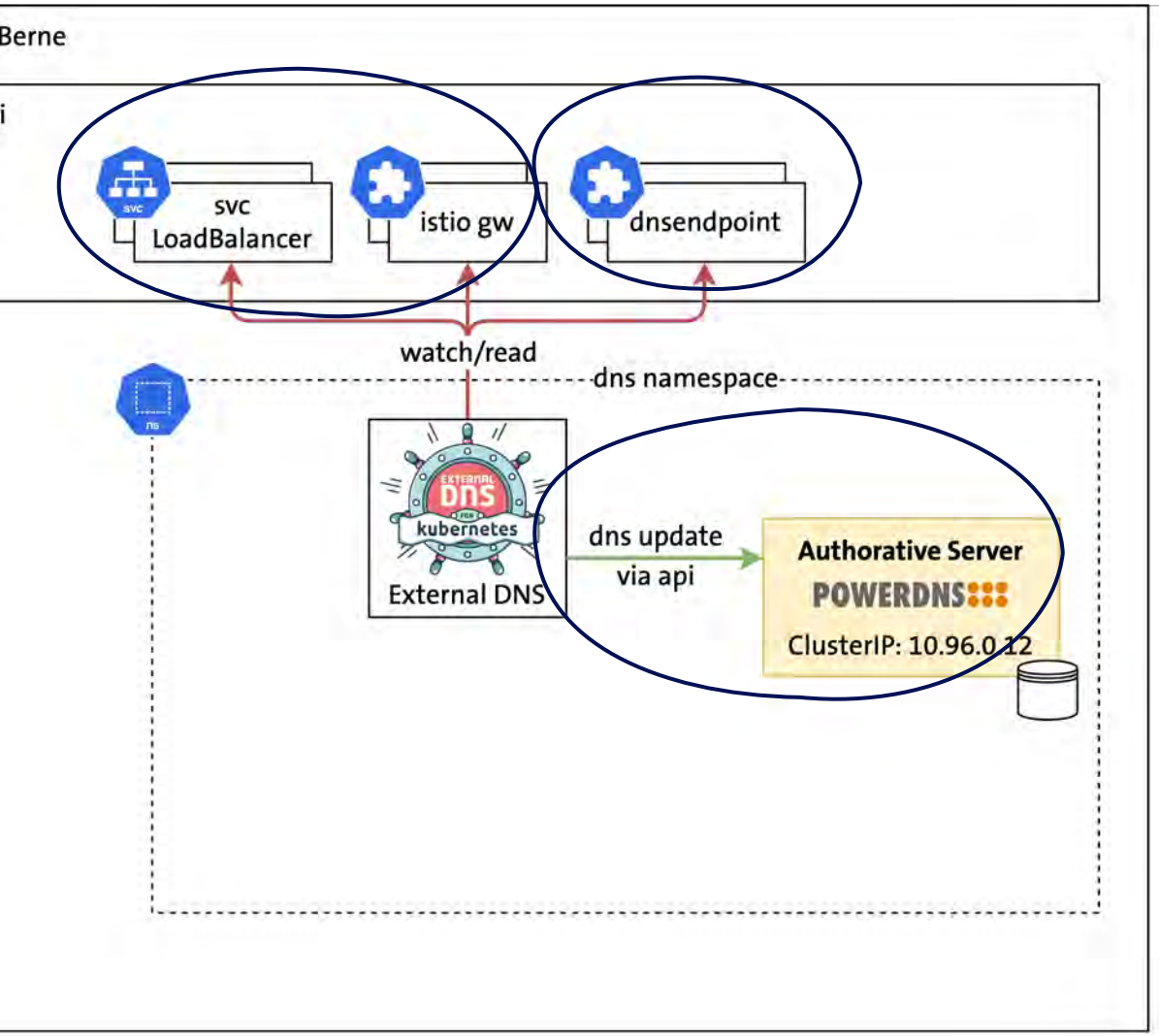
* https://github.com/kubernetes-sigs/external-dns
** After a fix in external-dns https://github.com/kubernetes-sigs/external-dns/pull/4212

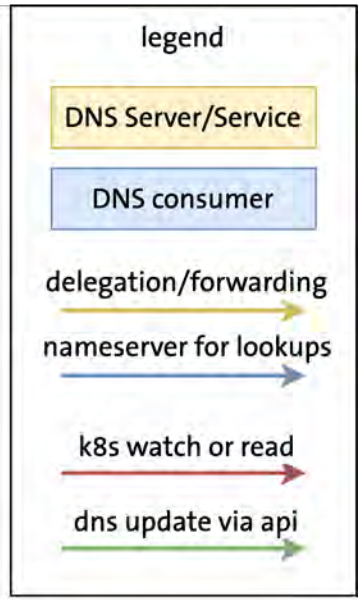

cluster site Berne
kube-api
pod
svc
ClusterIP
read
kube-system namespace
pod
(e.g. a 5G CNF)
dns query
node dns cache
CoreDNS
NodeIP: 172.18.0.3
default
forwarder
Kubernetes DNS
CoreDNS
ClusterIP: 10.96.0.10

Overview


cluster site Berne
kube-api
pod
svc
ClusterIP
svc
LoadBalancer
istio gw
dnsendpoint
watch/read
dns namespace
read
External DNS
dns update
via api
Authorative Server
POWERDNS
ClusterIP: 10.96.0.12
kube-system namespace
pod
(e.g. a 5G CNF)
dns query
node dns cache
CoreDNS
NodeIP: 172.18.0.3
default
forwarder
Kubernetes DNS
CoreDNS
ClusterIP: 10.96.0.10
legend
DNS Server/Service
DNS consumer
delegation/forwarding
nameserver for lookups
k8s watch or read
dns update via api

Automation of Authoritative Server Using ExternalDNS





ExternalDNS reads from kube-api:

- Static Resource Records as DNSEndpoint Custom Resources
- Dynamic Type A Records using Annotations
  - Name definition via Annotation
  - IP fetched from Service / Ingress / Istiogw status field

ExternalDNS writes to PowerDNS API

ExternalDNS State Management for Dynamic IP Assigment: GitOps + Kubernetes

```yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    external-dns.alpha.kubernetes.io/hostname: my-app.example.com
  name: my-app
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    name: my-app
  type: LoadBalancer
```

DNS Name defined in git

```yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    external-dns.alpha.kubernetes.io/hostname: my-app.example
  name: my-app
spec:
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: 80
  selector:
    name: my-app
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 192.168.0.35
```

IP read by ExternalDNS

DNS Backend:

```
my-app.example.com. 3600 IN A 192.168.0.35
```

ExternalDNS creates

ExternalDNS State Management for Static Assignment: GitOps

```yaml
apiVersion: externaldns.k8s.io/v1alpha1
kind: DNSEndpoint
metadata:
  name: my-mx-record
spec:
  endpoints:
    - dnsName: my-app.example.com
      recordTTL: 3600
      recordType: A
      targets:
        - 192.168.0.35
    - dnsName: example.com
      recordTTL: 3600
      recordType: MX
      targets:
        - 10 mailhost1.example.com
        - 20 mailhost2.example.com
```

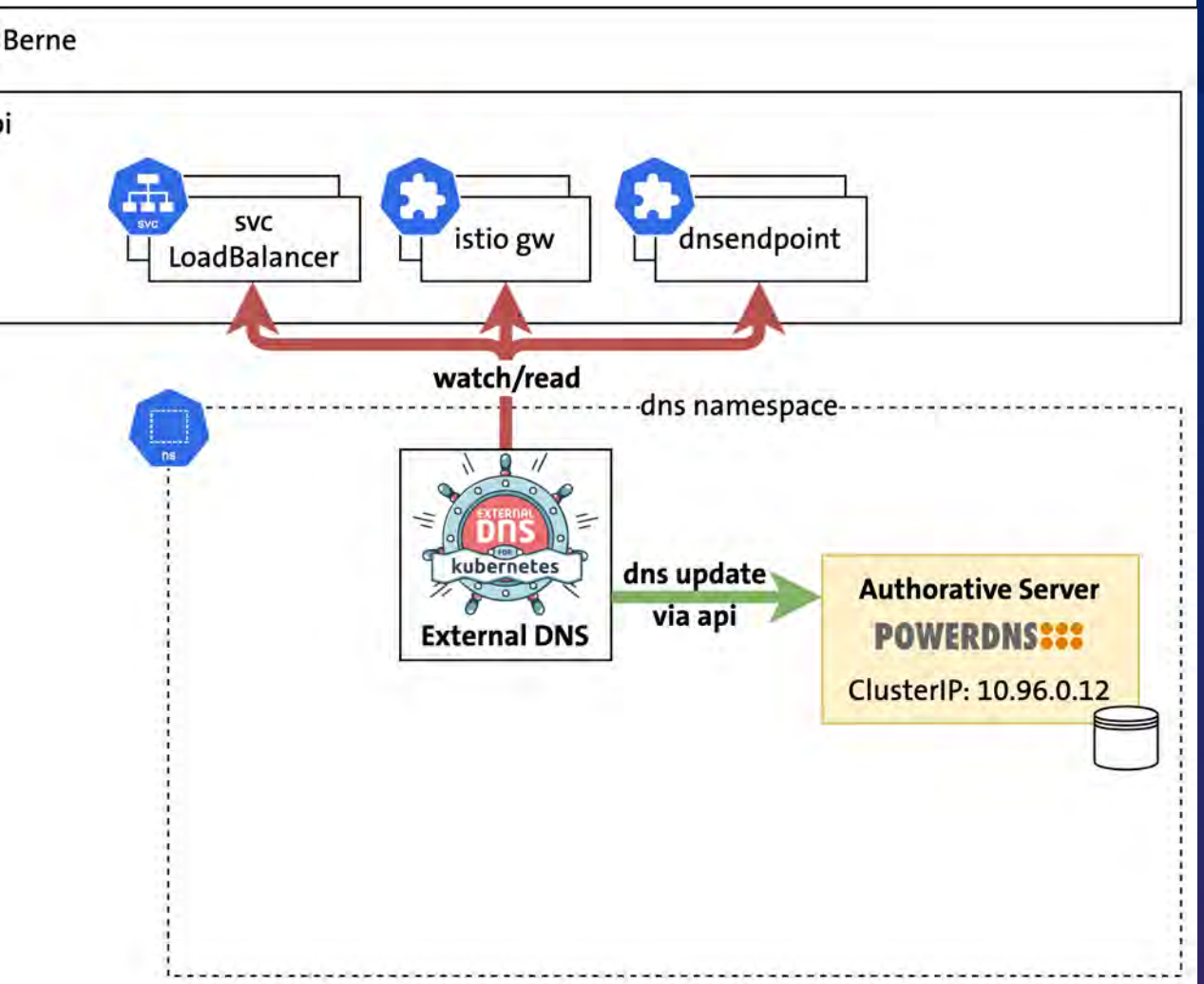
Resource Records defined in git

```yaml
apiVersion: externaldns.k8s.io/v1alpha1
kind: DNSEndpoint
metadata:
  name: my-mx-record
spec:
  endpoints:
    - dnsName: my-app.example.com
      recordTTL: 1
      recordType: A
      targets:
        - 192.168.0.35
    - dnsName: example.com
      recordTTL: 180
      recordType: MX
      targets:
        - 10 mailhost1.example.com
        - 20 mailhost2.example.com
```

ExternalDNS creates

DNS Backend:

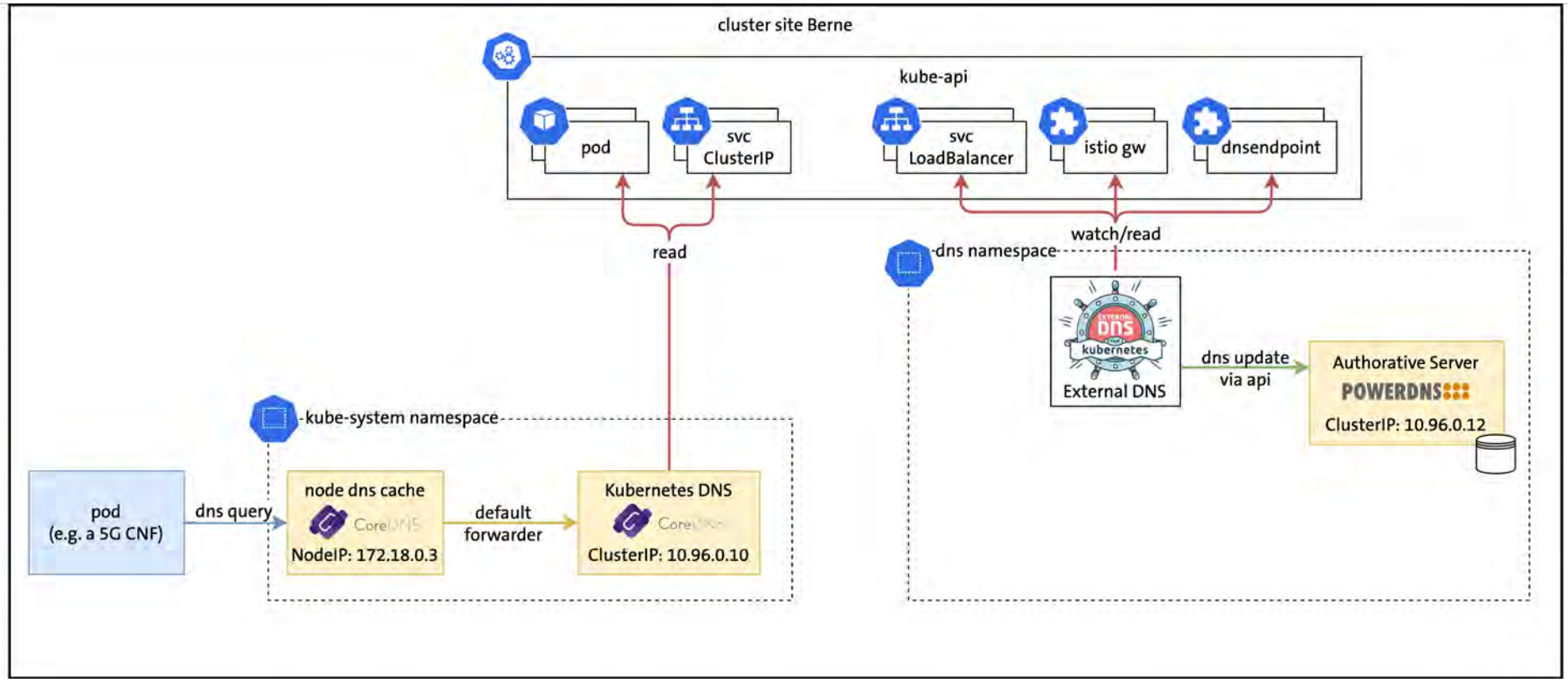
```
my-app.example.com.  3600 IN A  192.168.0.35
example.com.         3600 IN MX 10 mailhost1.example.com
example.com.         3600 IN MX 20 mailhost2.example.com
```



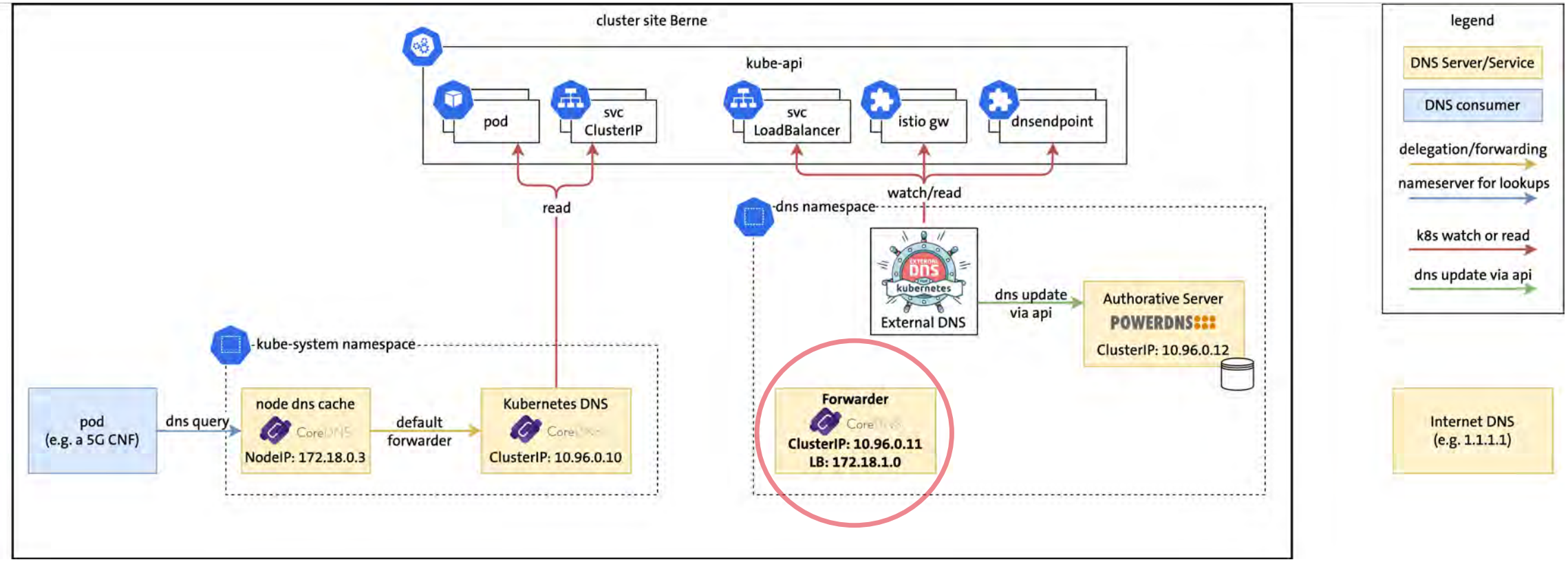
Demo ExternalDNS + PowerDNS Single Cluster

https://github.com/swisscom/cloud-native-telco/tree/main/prototypes/dns/1-demo-external-dns

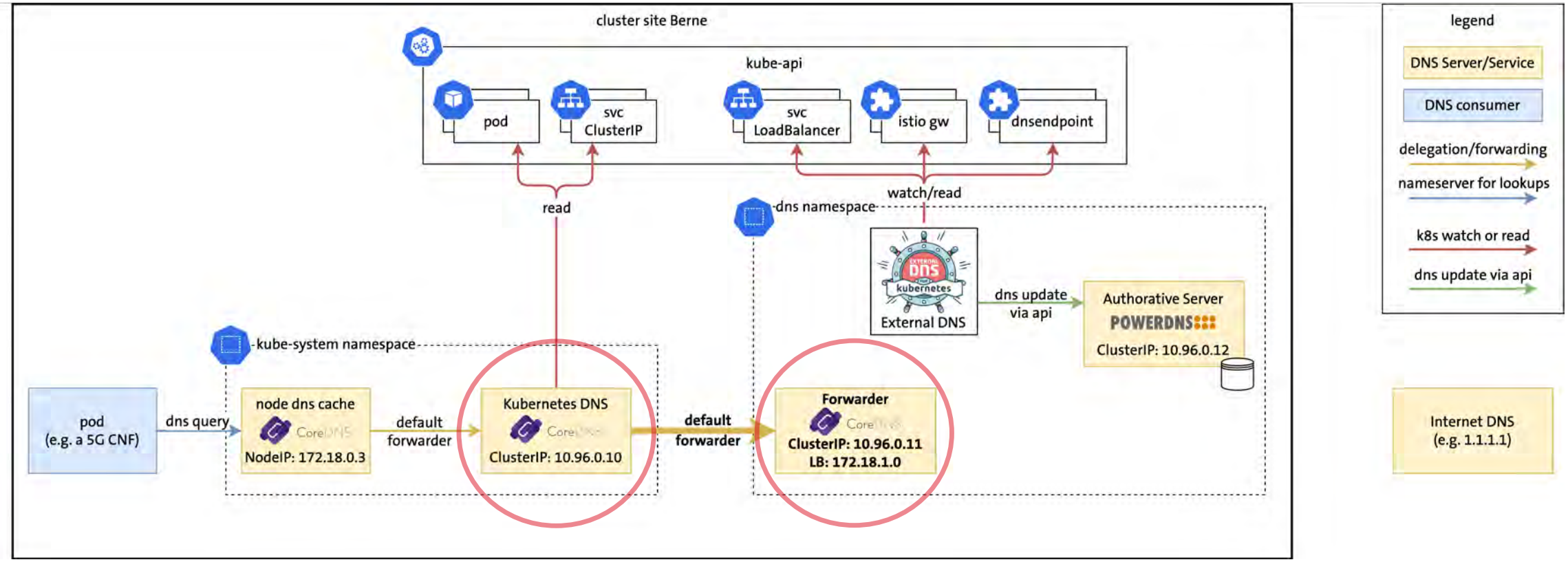
Forwarding to Authoritative Server



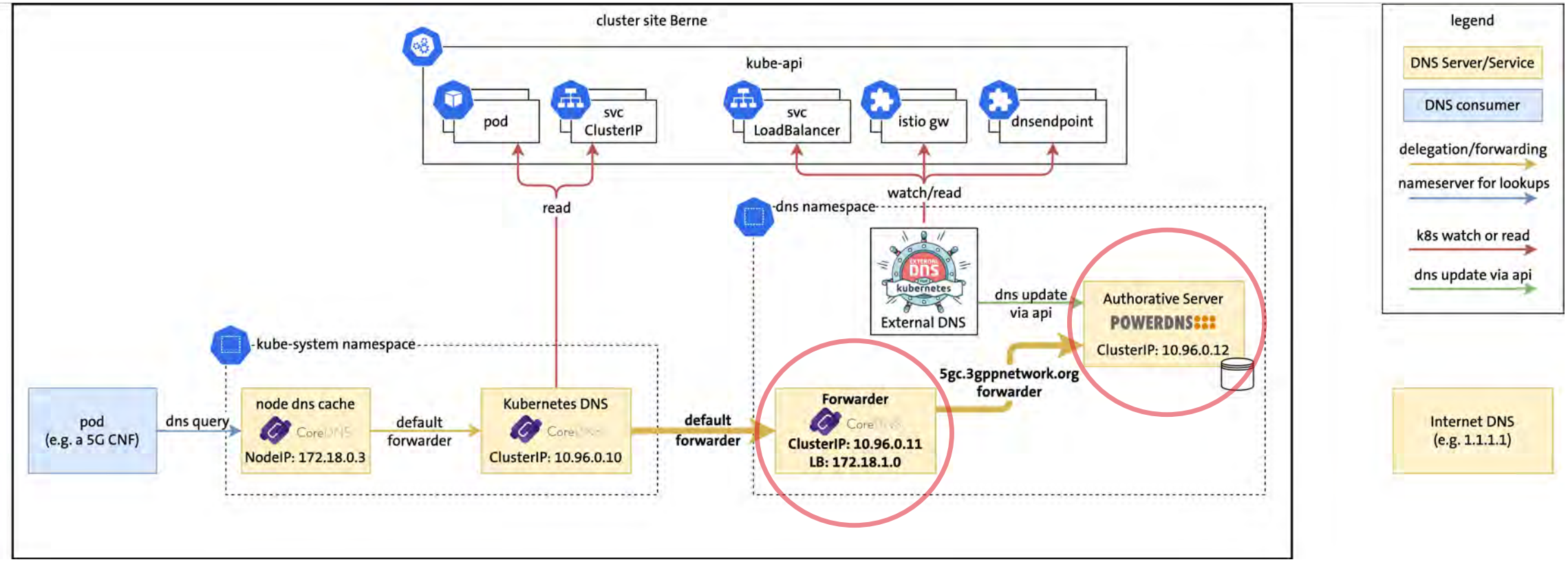
Forwarding to Authoritative Server



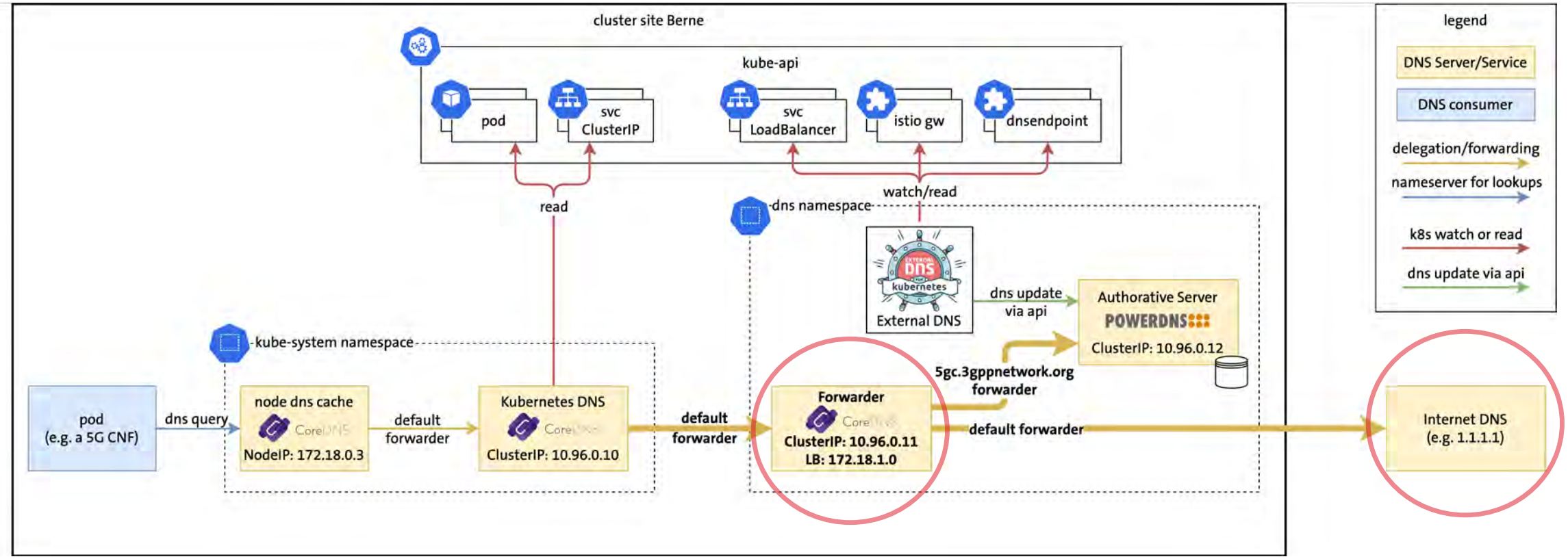
Forwarding to Authoritative Server



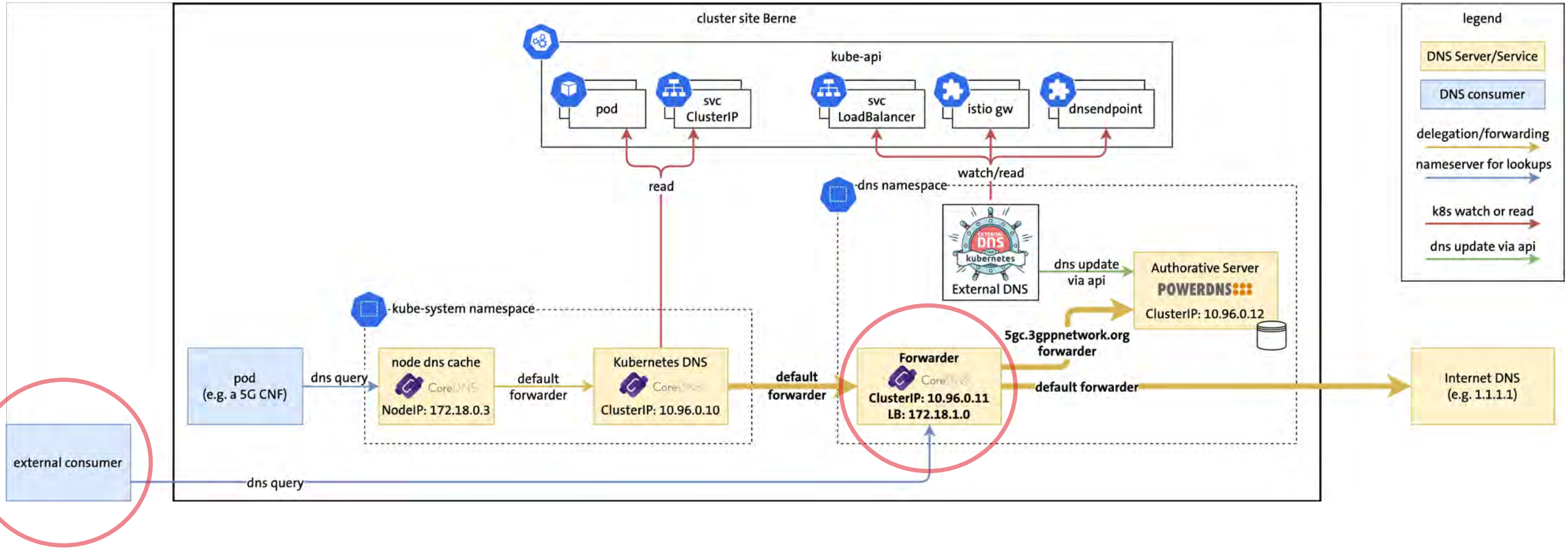
Forwarding to Authoritative Server



Forwarding to Authoritative Server

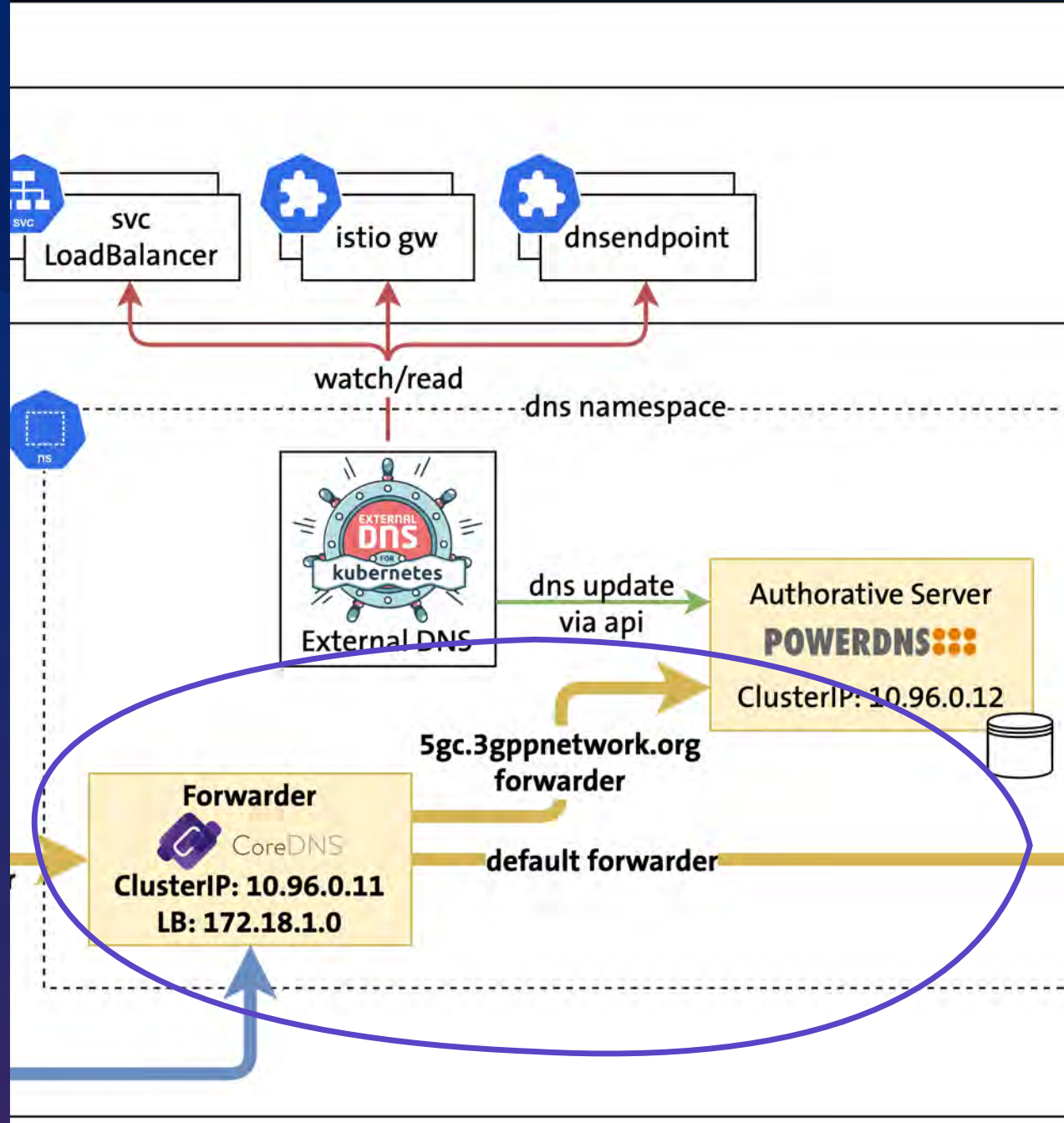


Forwarding to Authoritative Server

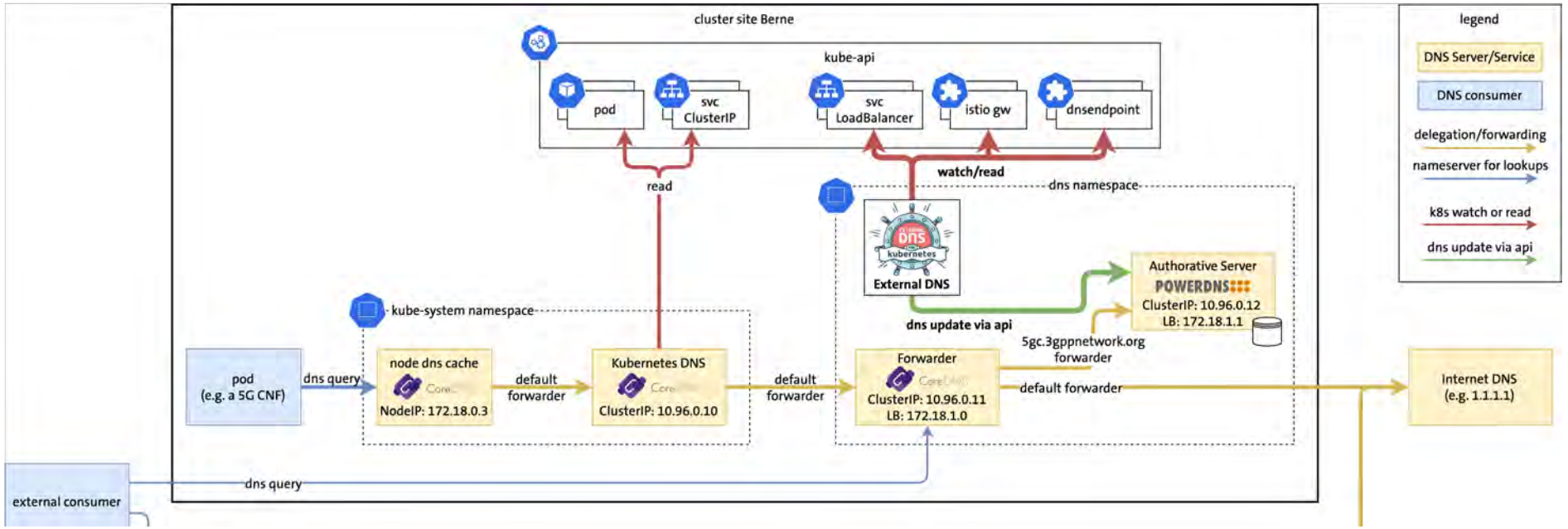


Demo Forwarding

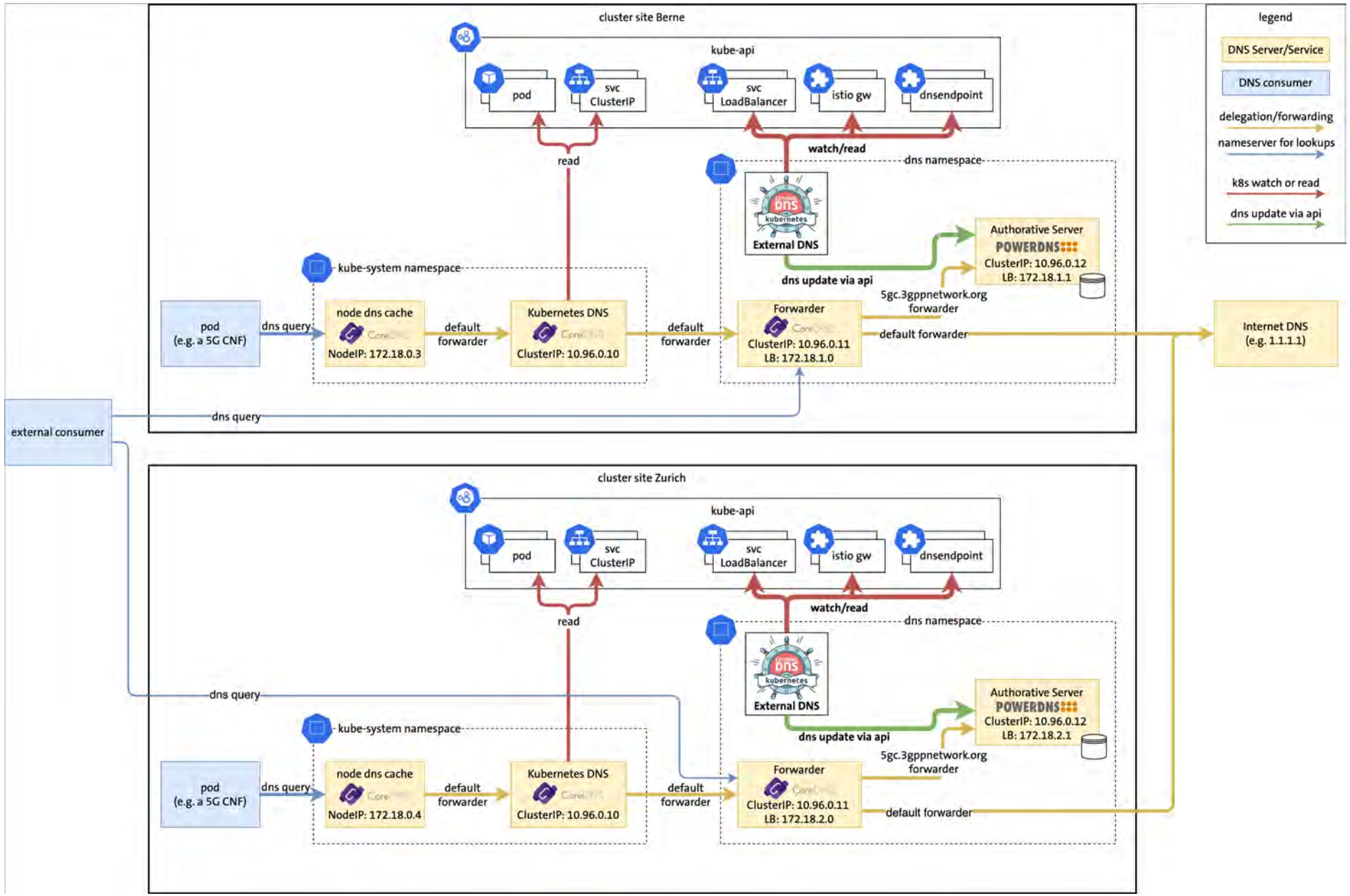
https://github.com/swisscom/cloud-native-telco/tree/main/prototypes/dns/2-demo-forwarding



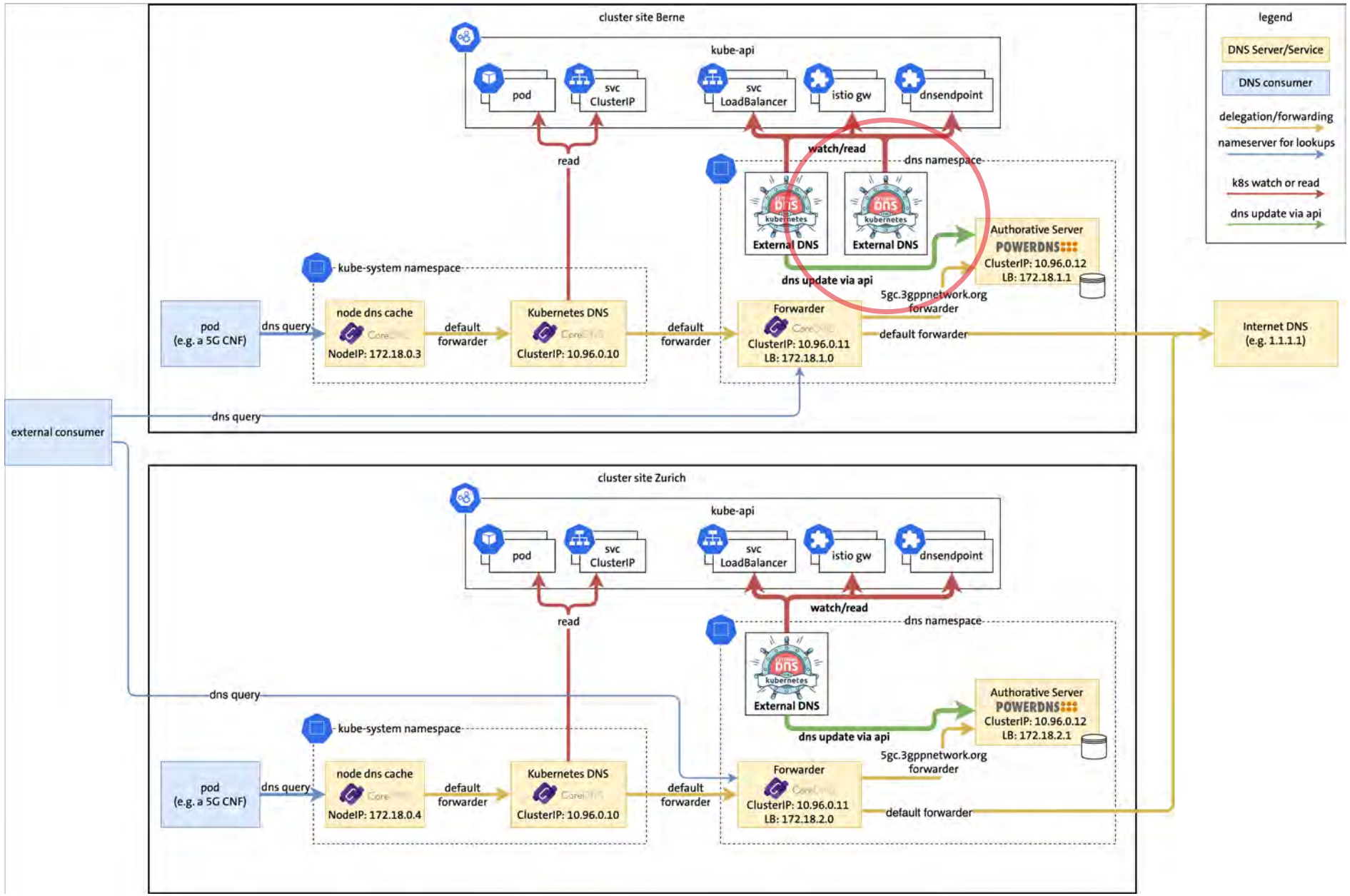
Dual-Cluster Using ExternalDNS



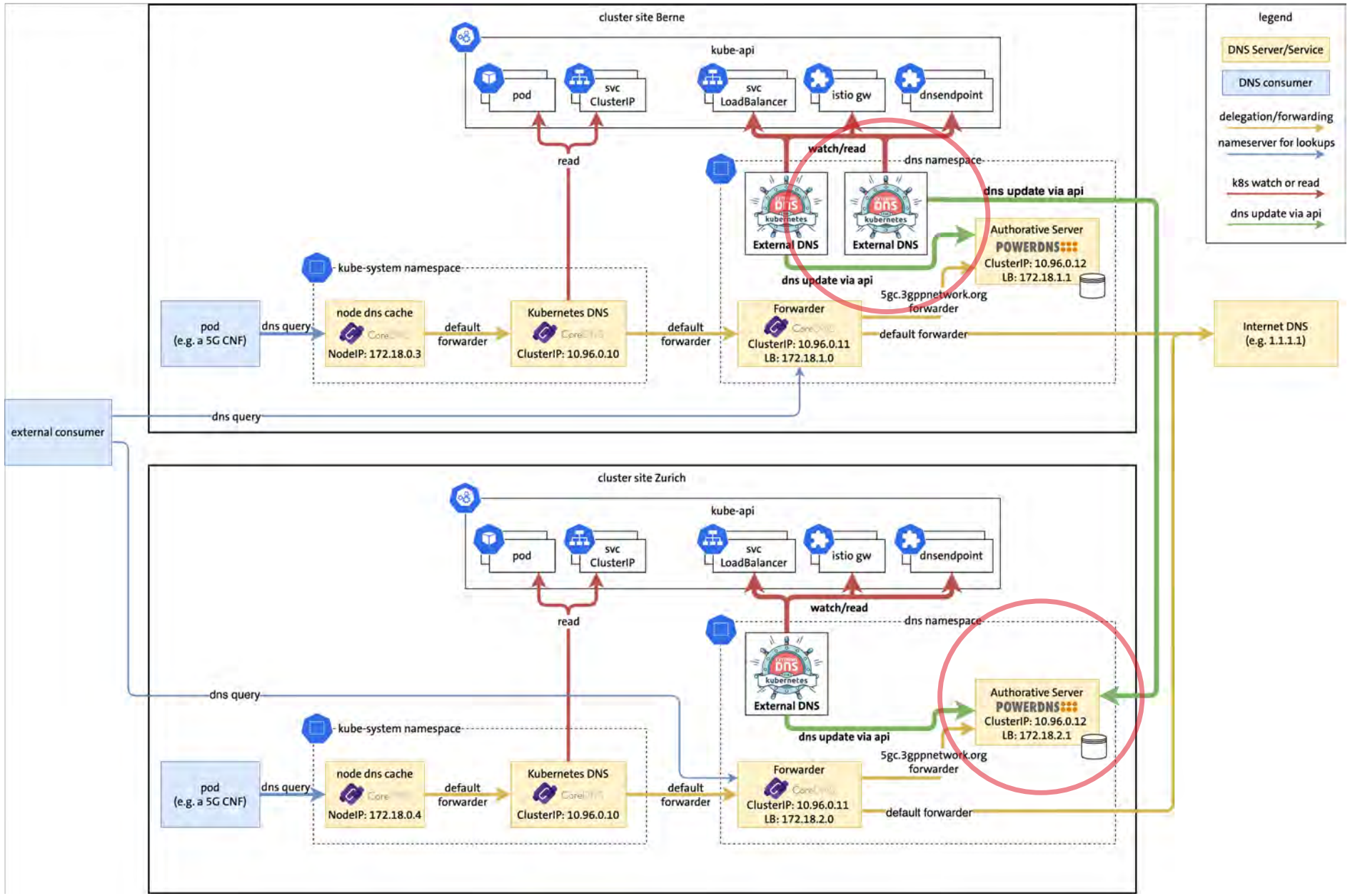
Dual-Cluster Using ExternalDNS



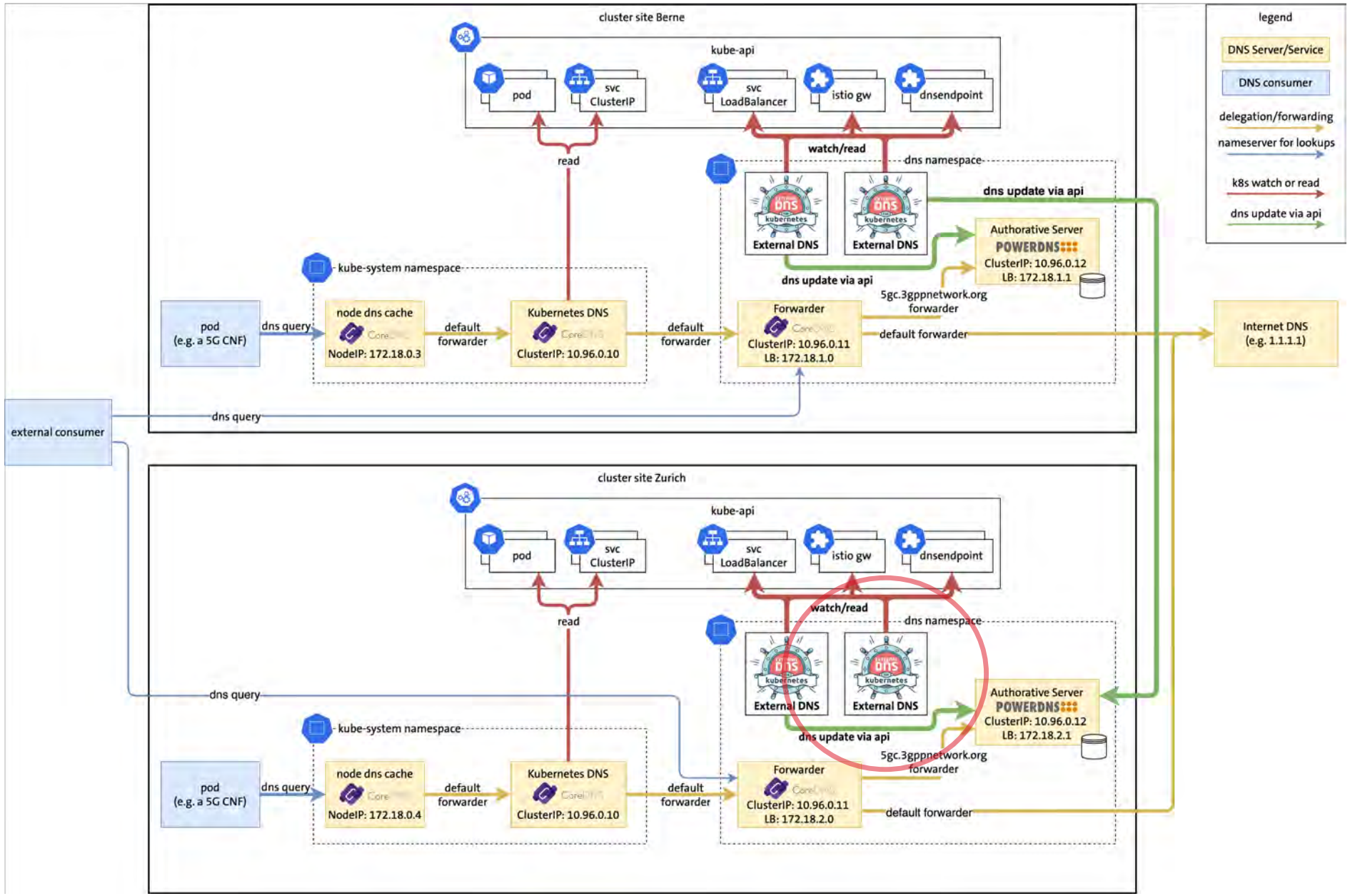
Dual-Cluster Using ExternalDNS



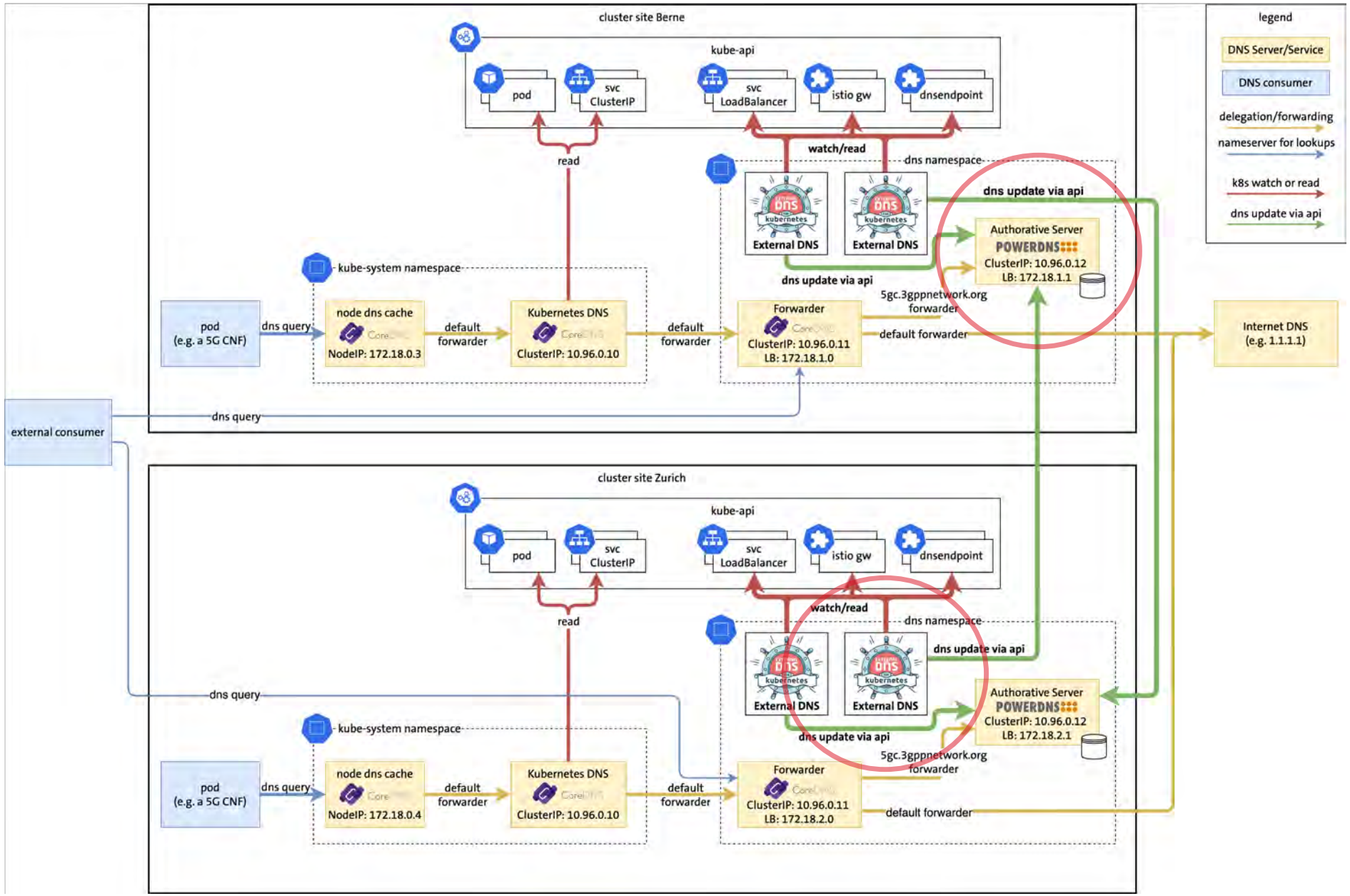
Dual-Cluster Using ExternalDNS



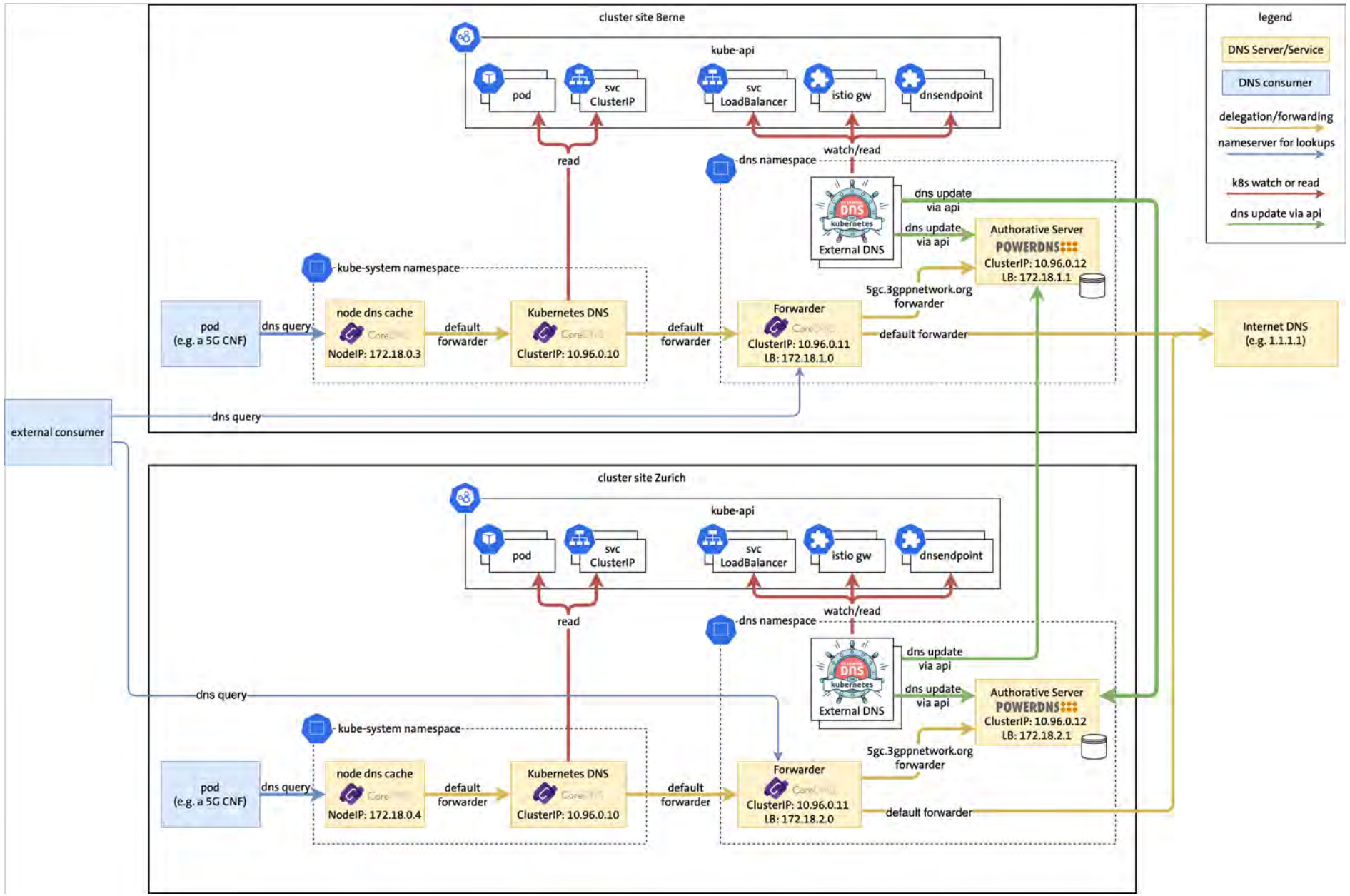
Dual-Cluster Using ExternalDNS



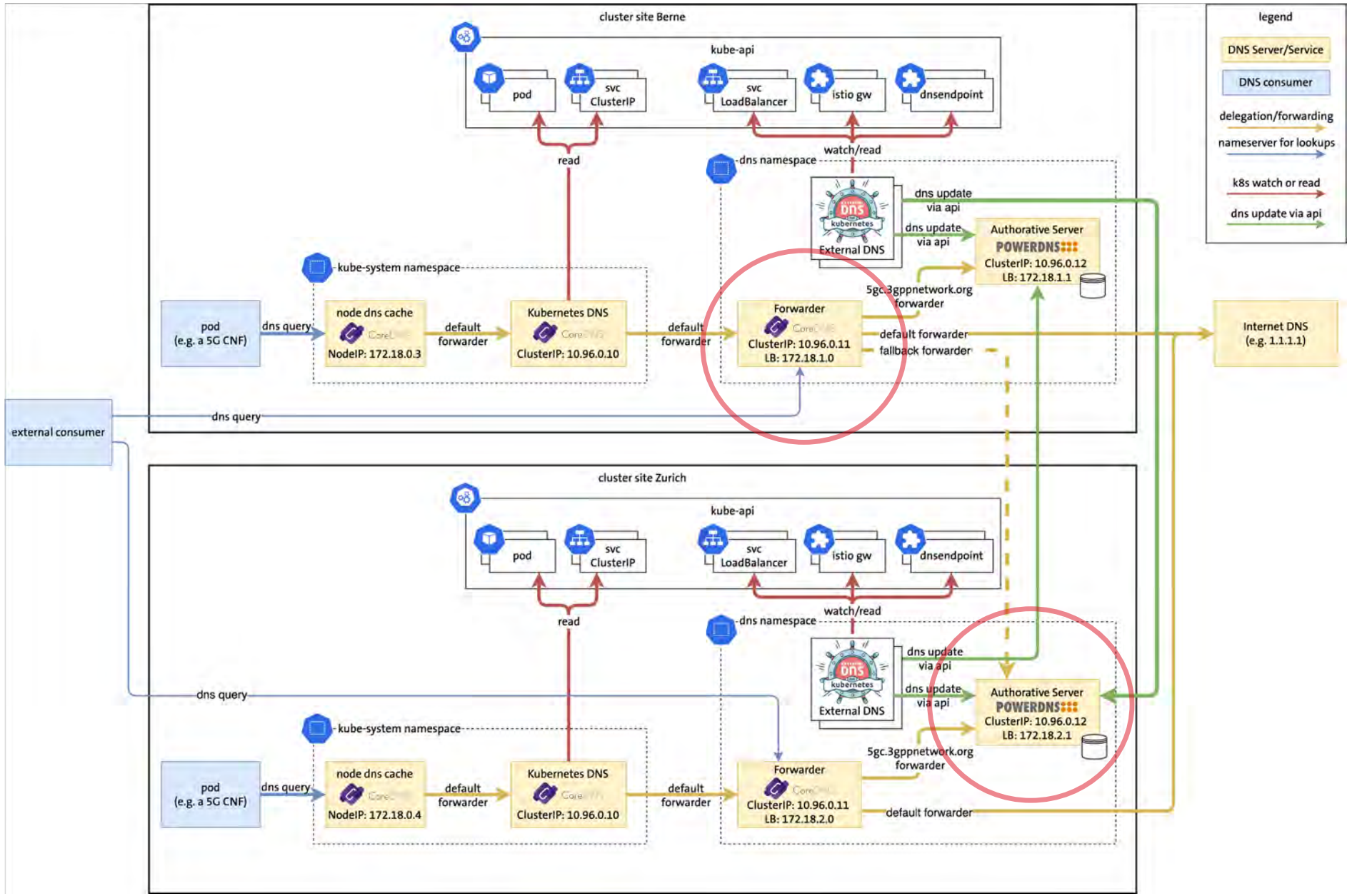
Dual-Cluster Using ExternalDNS



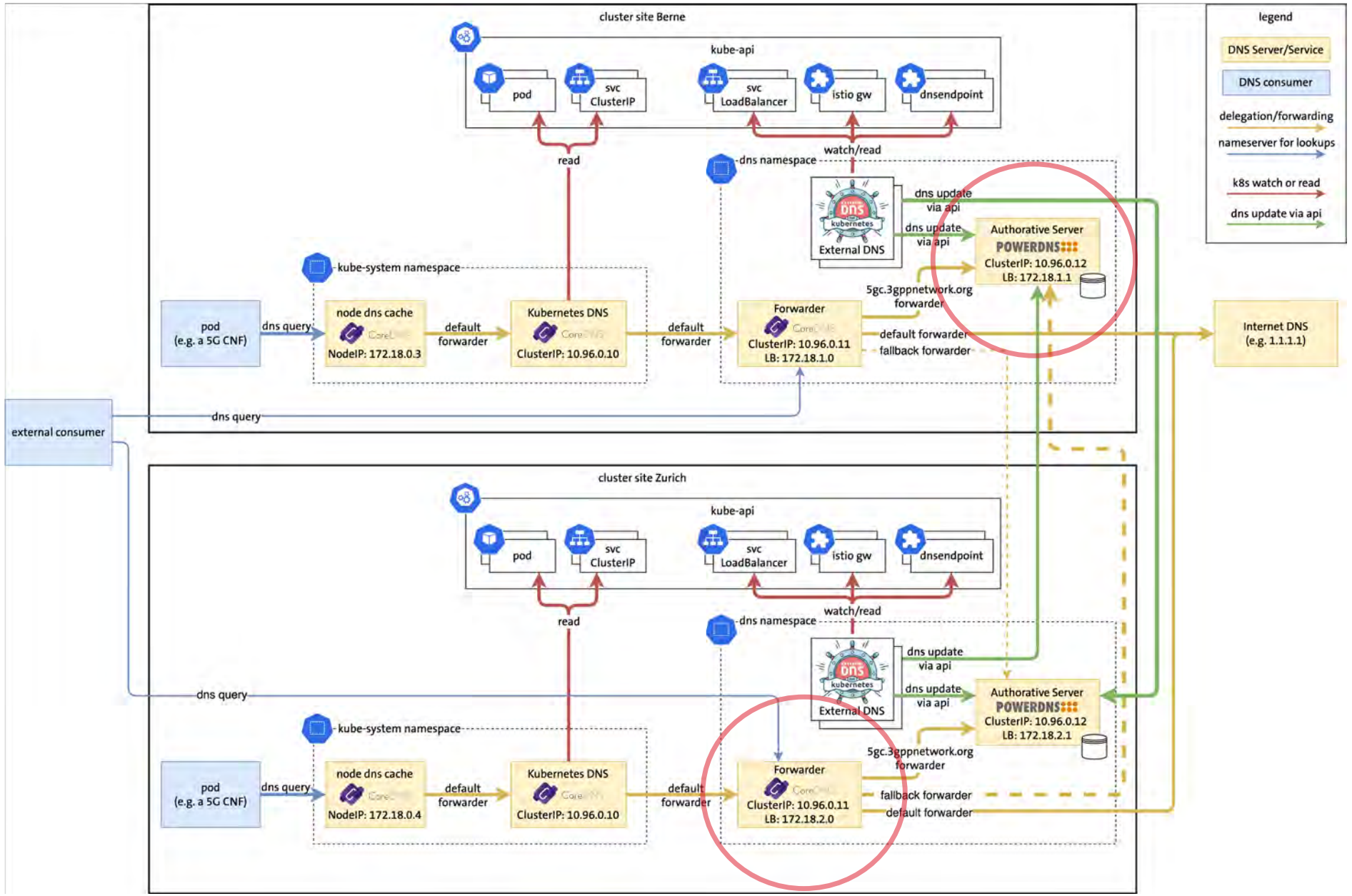
Dual-Cluster Using ExternalDNS



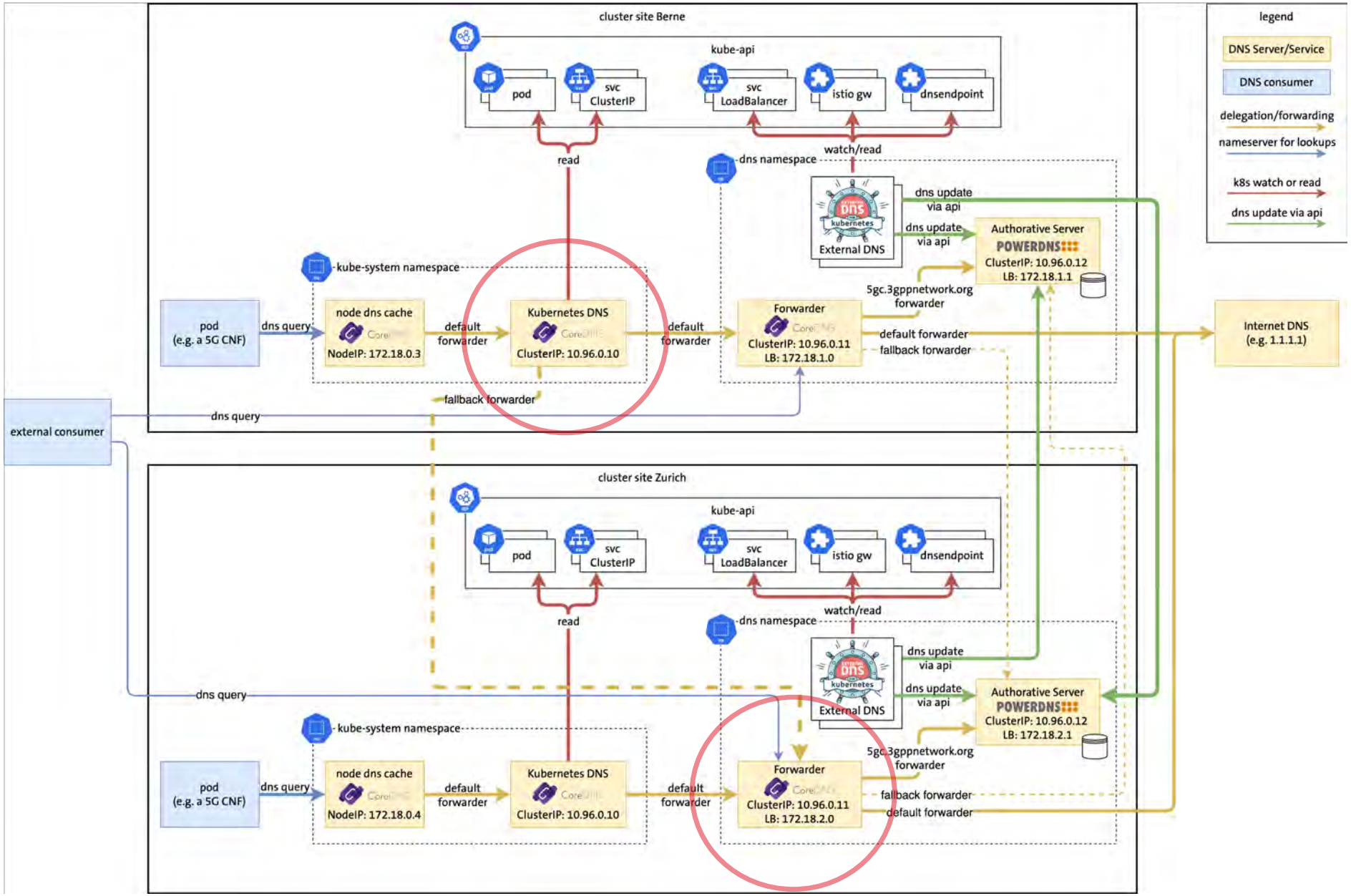
Eliminating Single Point of Failure



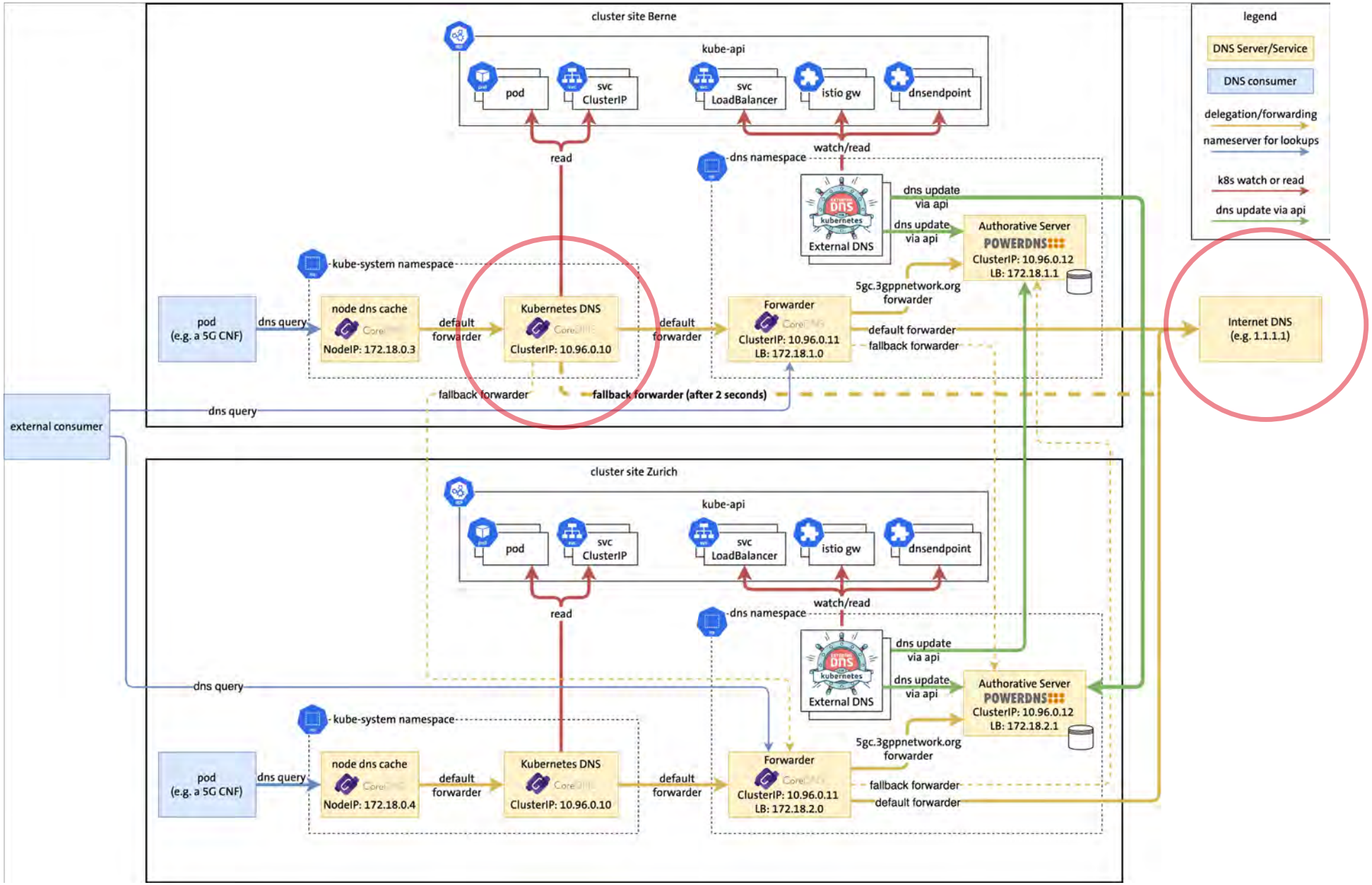
Eliminating Single Point of Failure



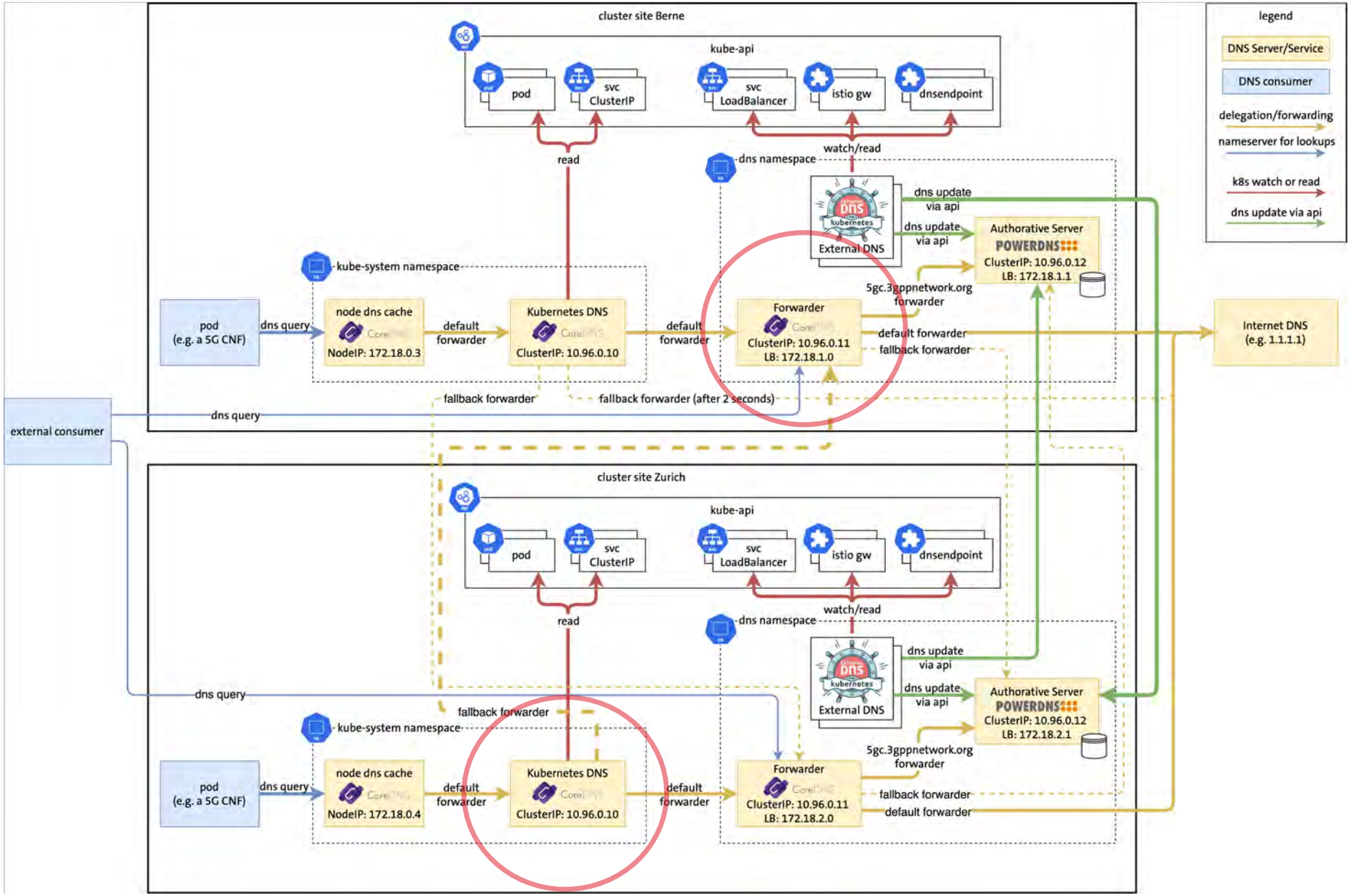
Eliminating Single Point of Failure



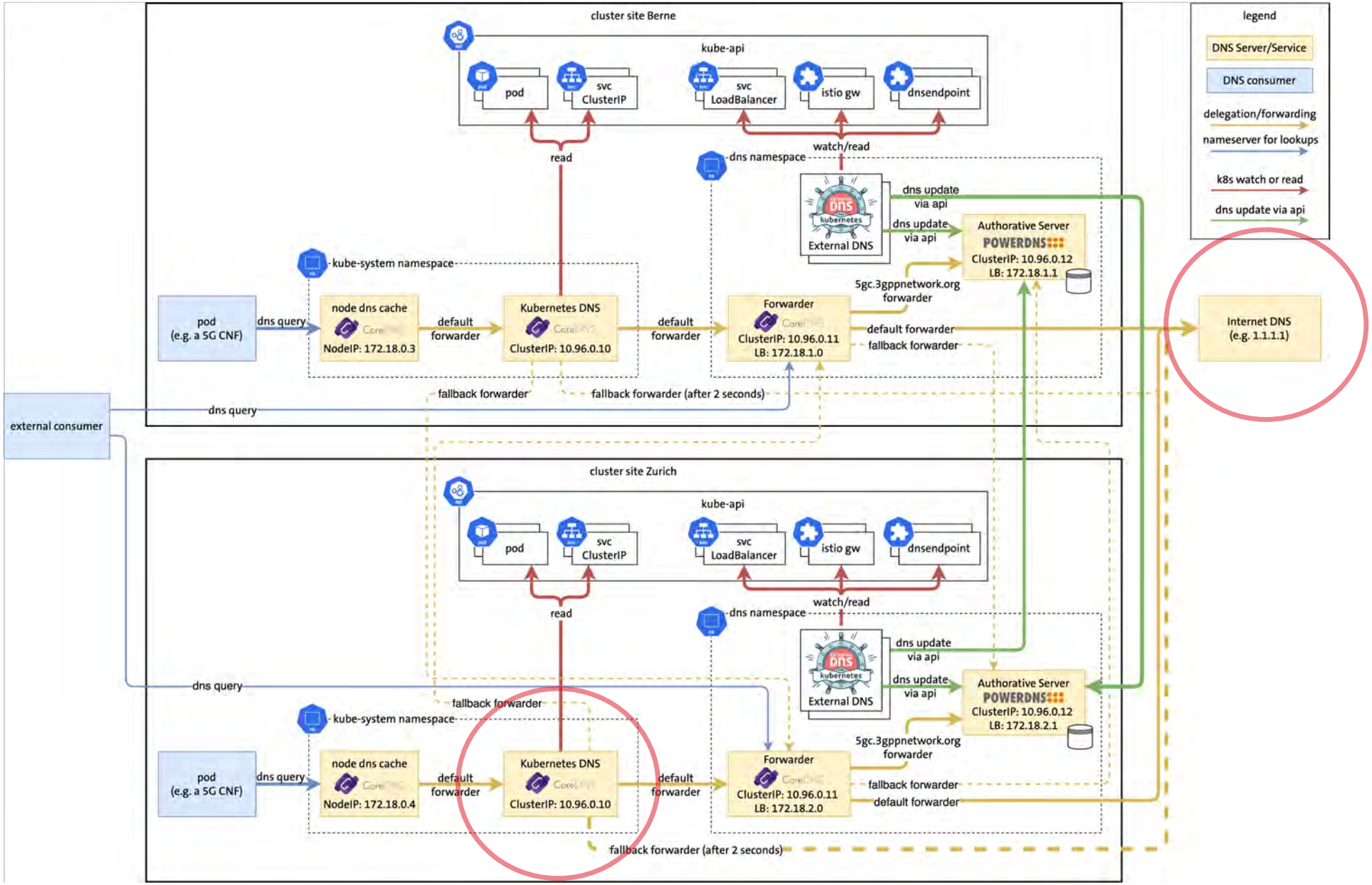
Eliminating Single Point of Failure



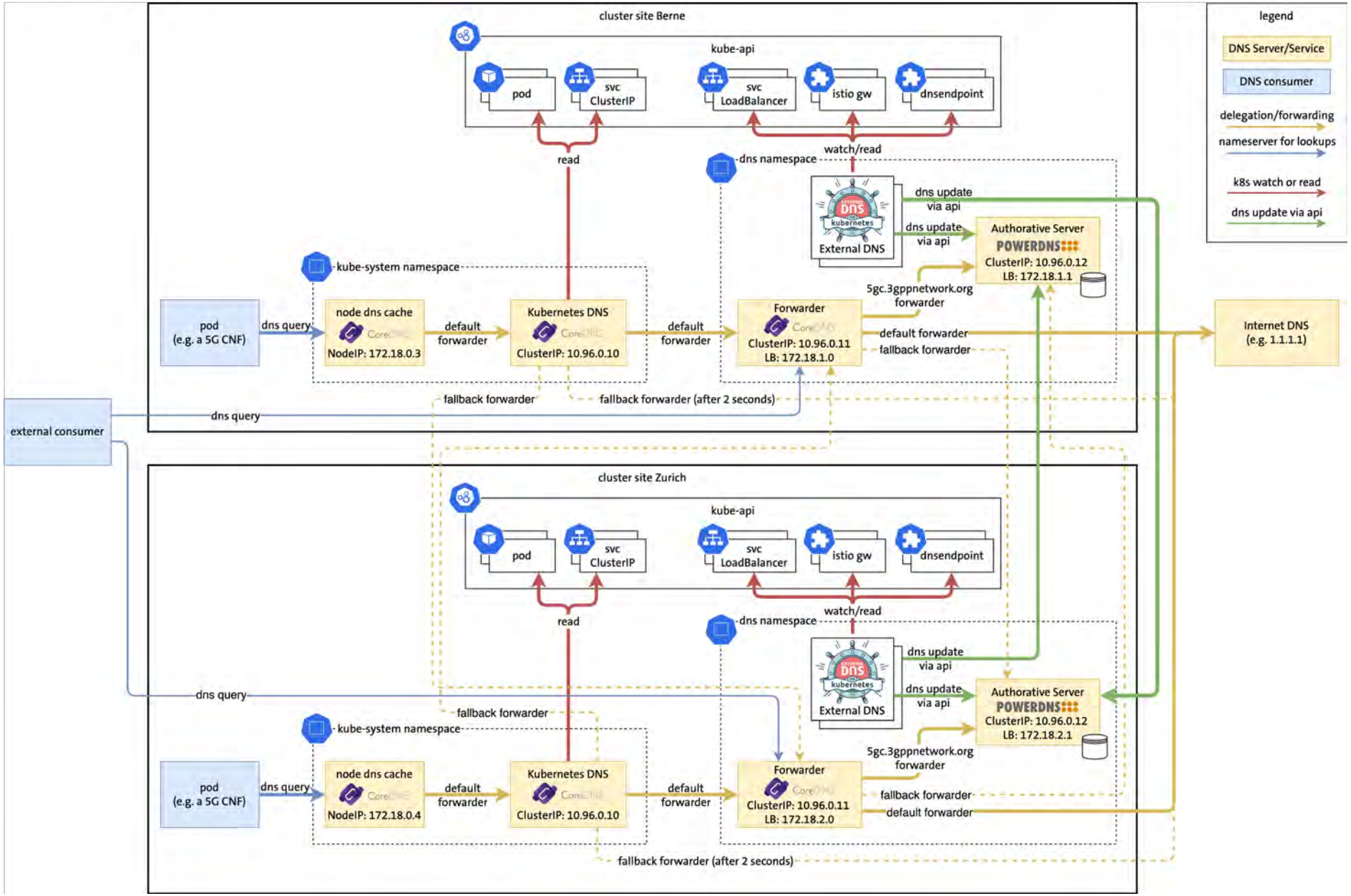
Eliminating Single Point of Failure



Eliminating Single Point of Failure

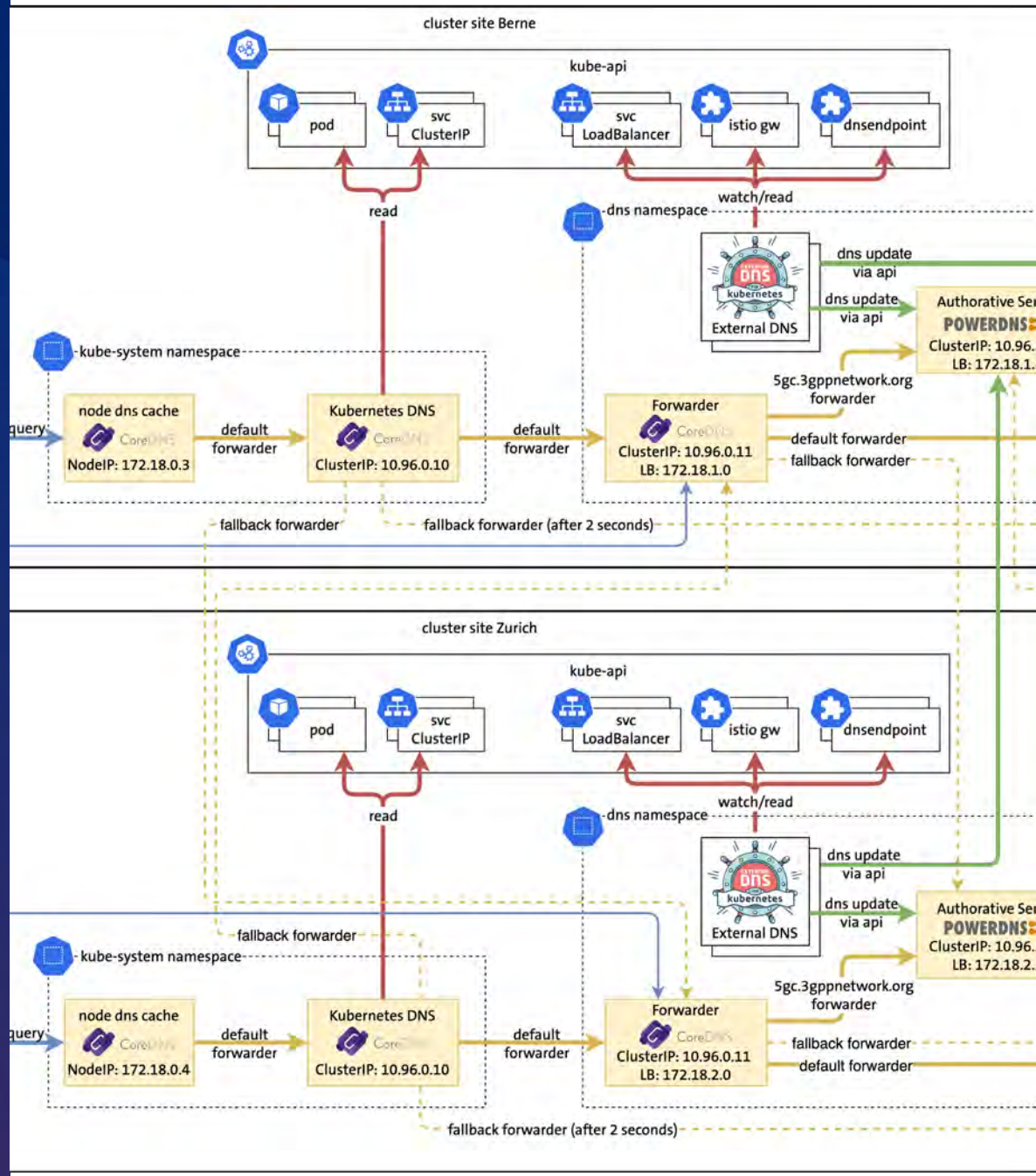


Final Dual Cluster Setup



Demo Multi Cluster

https://github.com/swisscom/cloud-native-telco/tree/main/prototypes/dns/3-demo-multi-cluster-dns



Limitations of Our DNS Service

Self-dependence

Complexity increases when consuming the Service from within the same clusters.

Kubernetes Resources only

Limited to Kubernetes Resources and GitOps

Service Discovery

ExternalDNS not suited for service discovery

Limitations of ExternalDNS: Service Discovery

Interval-Based Syncing due to architectural decisions

⚠️ Delayed Resource Record creation

No Health Checks (e.g. integration into Kubernetes Services/EndpointSlices)

⚠️ Cannot rely on ExternalDNS for app readiness

No Multi Cluster Round Robin for A records: one record cannot be shared by multiple ExternalDNS

⚠️ Cannot use DNS records created by ExternalDNS for routing across multiple clusters

Full cluster outage will not revoke DNS records

⚠️ Tight monitoring and additional automation needed to avoid outages

What Did We Achieve?

Proximity to Consumer
Minimal amount of hops between 5G Core and DNS

✓ On-prem deployment

Fully Automated
GitOps driven and automated provisioning of DNS records

✓ GitOps + ExternalDNS

Geo Redundant & HA
Spread across multiple K8s clusters and geo regions to increase reliability

✓ Spread across multiple K8s Clusters

Support of Advanced DNS features
Resource Records such as NAPTR and SRV supported for e.g. SIP Phone Calls

✓ Advanced RRs supported

K8s integration with ExternalDNS
The System leverages Kubernetes Patterns such as CRs and Operators

✓ 100% Kubernetes Resources

Minimal Amount of SPOFs
Remove single points of failure from the System

✓ Distributed control plane
✓ Cross-Cluster Forwarding


Thanks!

Resilience Testing on Cloud Native DNS

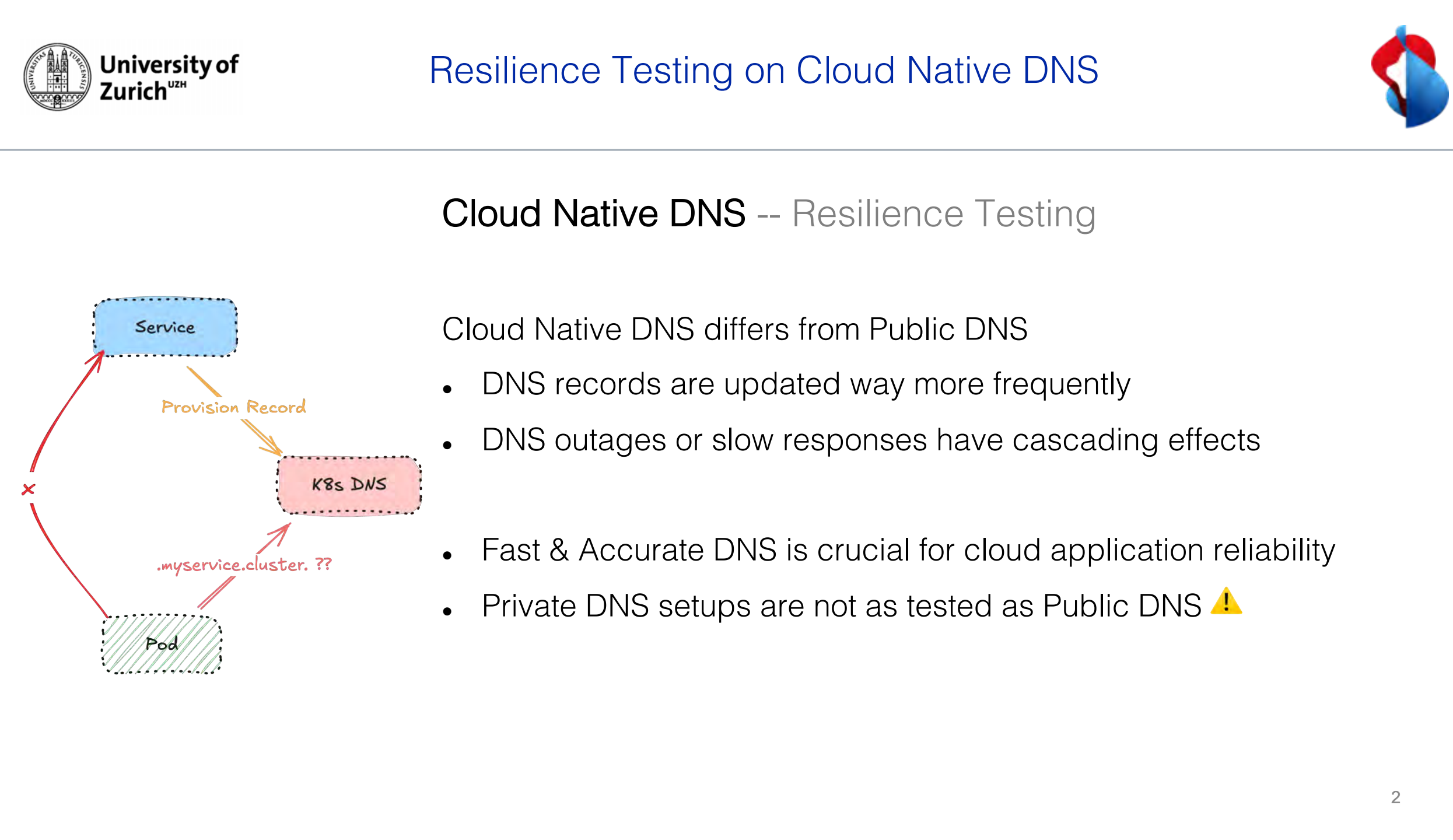
Georgios Daskalopoulos

Swisscom

- Fabian Schulz
- Joel Studler

Communications Systems Lab @ UZH

- Dr. Burkhard Stiller
- Thomas Grübl

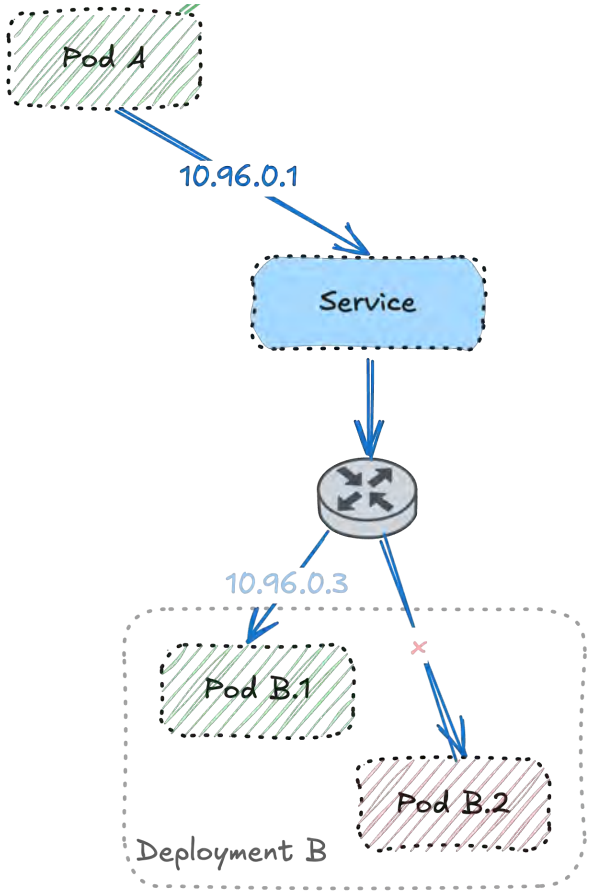


Cloud Native DNS -- Resilience Testing

Cloud Native DNS differs from Public DNS

- DNS records are updated way more frequently
- DNS outages or slow responses have cascading effects

- Fast & Accurate DNS is crucial for cloud application reliability
- Private DNS setups are not as tested as Public DNS ⚠️



Cloud Native DNS -- **Resilience Testing**

- Cloud Native DNS can be tested with Resilience Testing
- Resilience
  - Ability of an application to withstand and recover from failures in its components.
  - Achieved with multiple instances of the same Process

- Resilience Testing / Chaos Engineering

  Controlled experiments which assess a system's Resilience.
  - Define Stady State based on KPIs
  - Introduce chaos, process crashes, network disruptions
  - Measure the impact on the system behavior

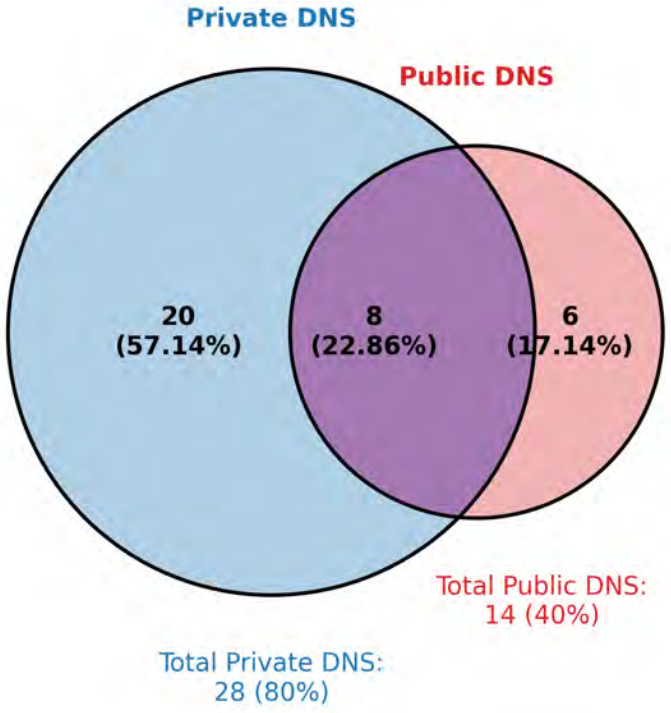
Real World Incidents

Source	Incidents
Google Cloud	19
AWS	7
Azure	4
Post-Mortem List	5
Total	35

35 Post - Mortems analysed from cloud providers

- Affected DNS Infrastructure
- DNS being the Root Cause
- Name Resolution
- Record Provisioning affected
- Public or Private Zone affected

Real World Incidents



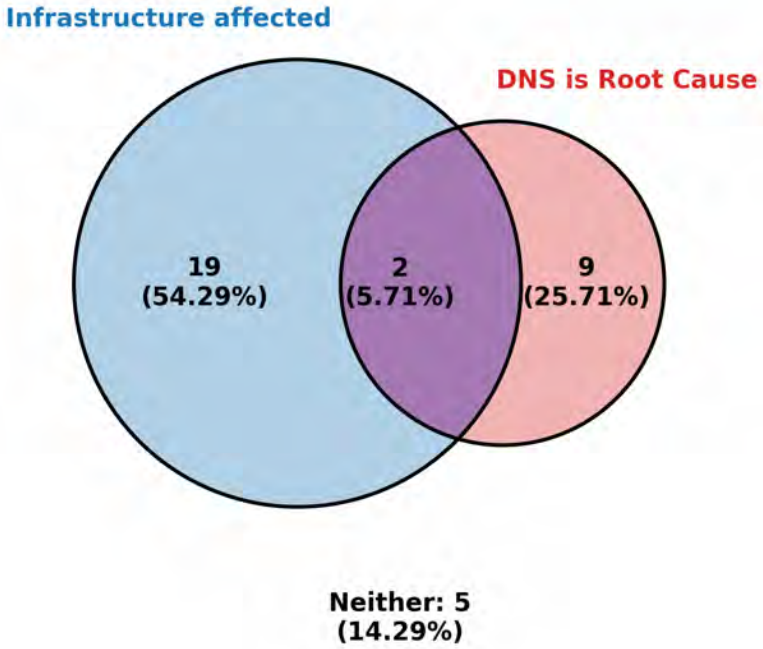
Incident Impact on DNS Zones

80% of incidents related to private DNS zones

57% of incidents related **only** to private DNS zones

Private DNS is involved in the majority of incidents

Real World Incidents



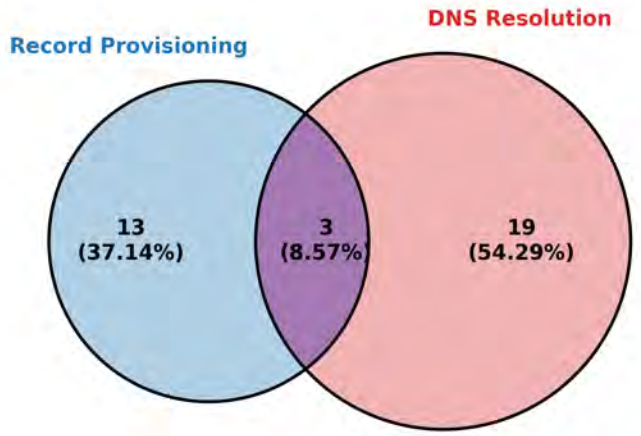
DNS Infrastructure Impact & DNS Root Cause Classification

54% of the incidents, DNS is impacted by infrastructure problems

25% of the incidents DNS is the root cause

DNS should withstand failure in its underlying components

Real World Incidents



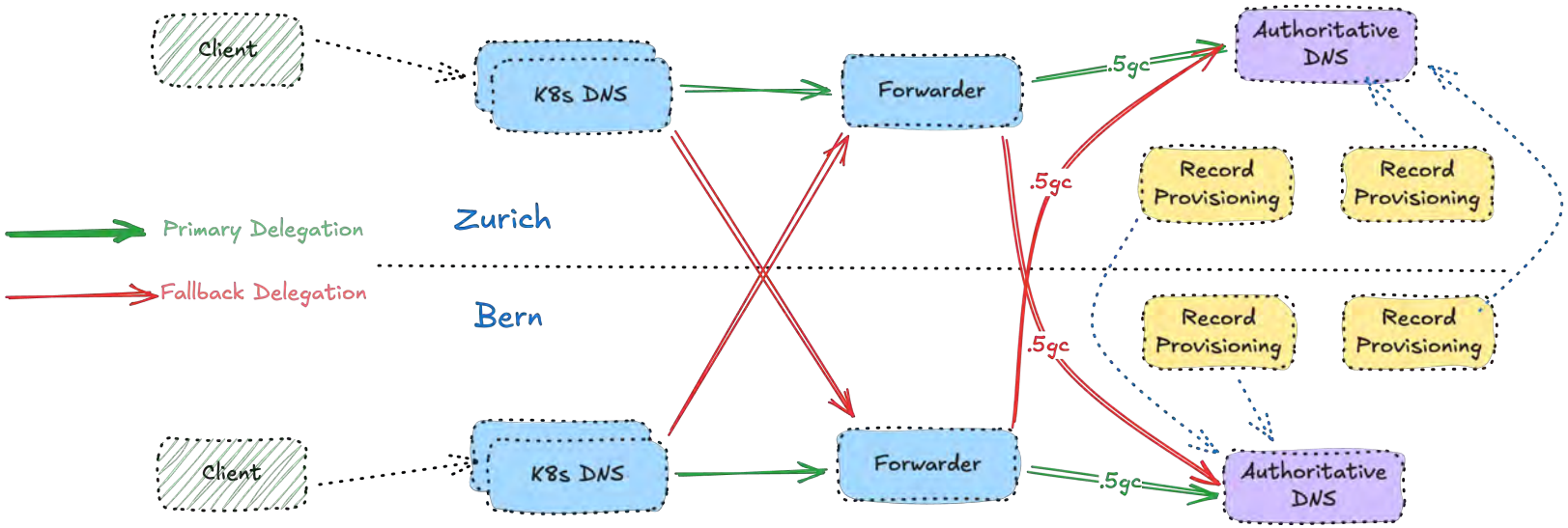
Incident Impact on DNS Functionality

54% of incidents, DNS Resolution is affected

37% of incidents, Record Provisioning is affected

No function is immune to disruption and need to be tested.

How → DNS Setup -- Testing Strategy – Metrics

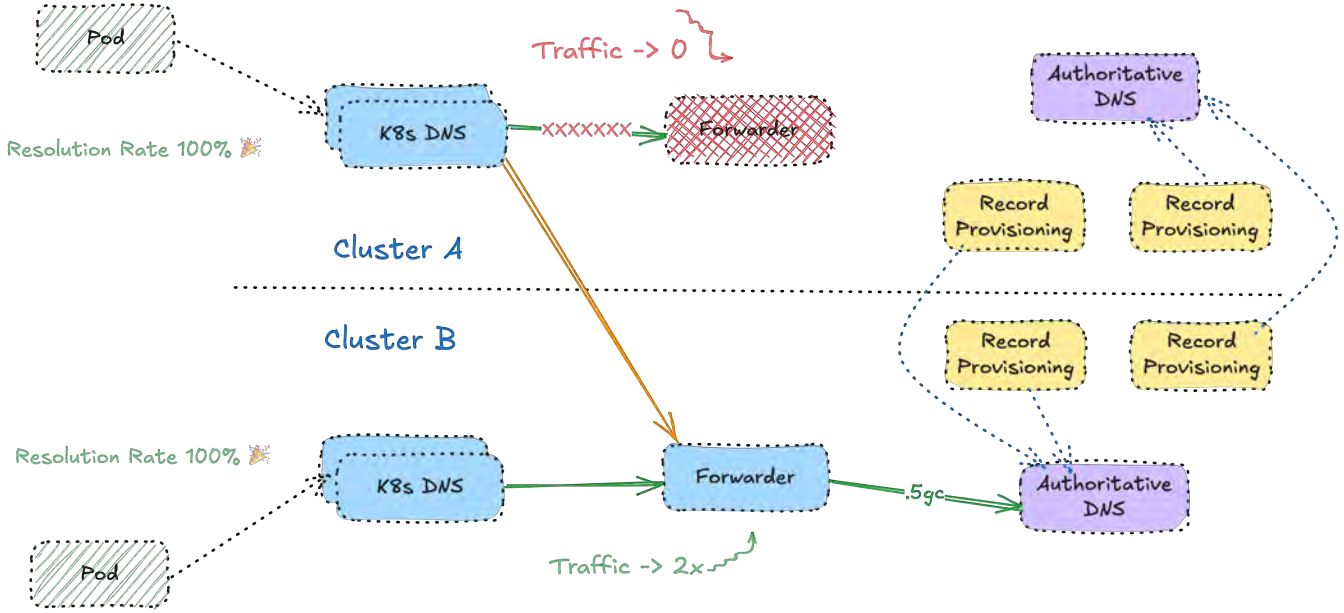


- Swisscom DNS Setup
- Multi-cluster DNS Resolution
- 5G Core Zone

- Primary & Fallback delegation
- Github Project

Resilience Testing on Cloud Native DNS

How → DNS Setup -- **Testing Strategy** -- Metrics



Chaos Injection, Pod Disruption:

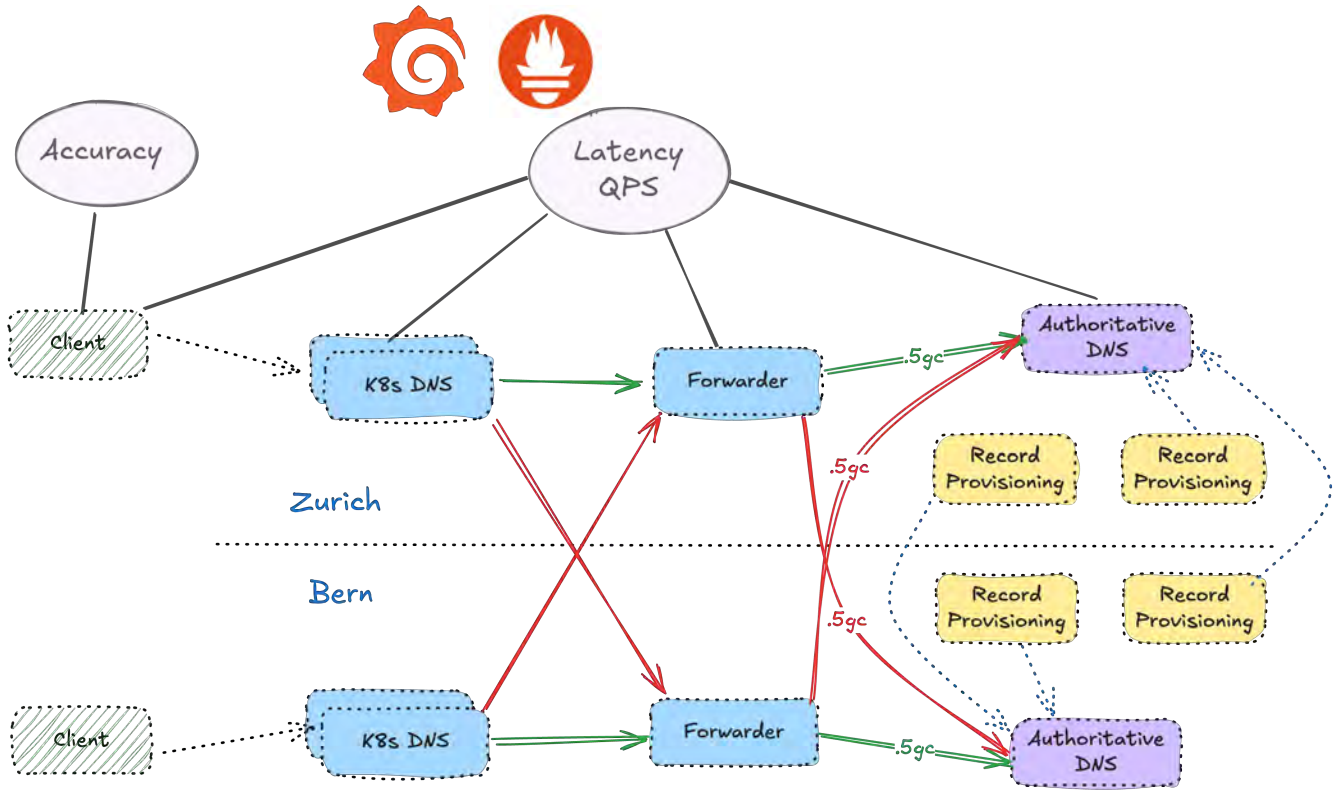
- Components on Resolution Path TC1, TC2, TC3
- Record Provisioning component TC4

DNS Traffic per Cluster

- 50 QPS
- Record Update per 10 seconds

Resilience Testing on Cloud Native DNS

How → DNS Setup -- Testing Strategy -- **Metrics**



Metrics Collected:

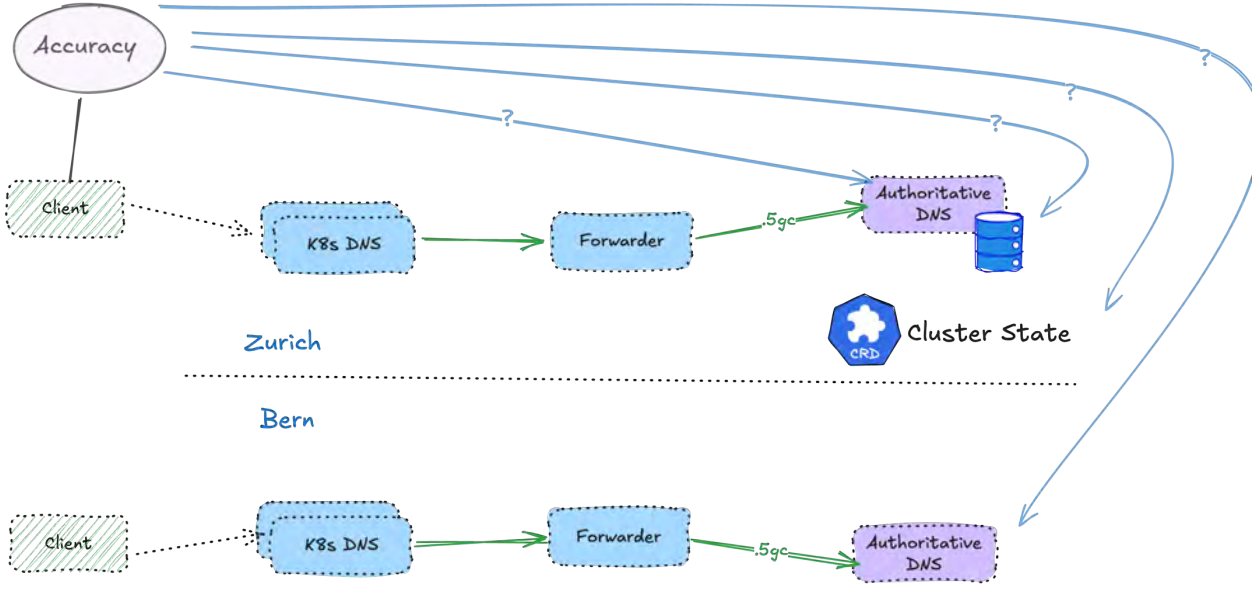
- Latency
- Queries Per Second
- Accuracy of Resolutions %

Observability Stack:

- Separate Kind Cluster
- Prometheus
- Grafana

Resilience Testing on Cloud Native DNS

How → Assessing Accuracy



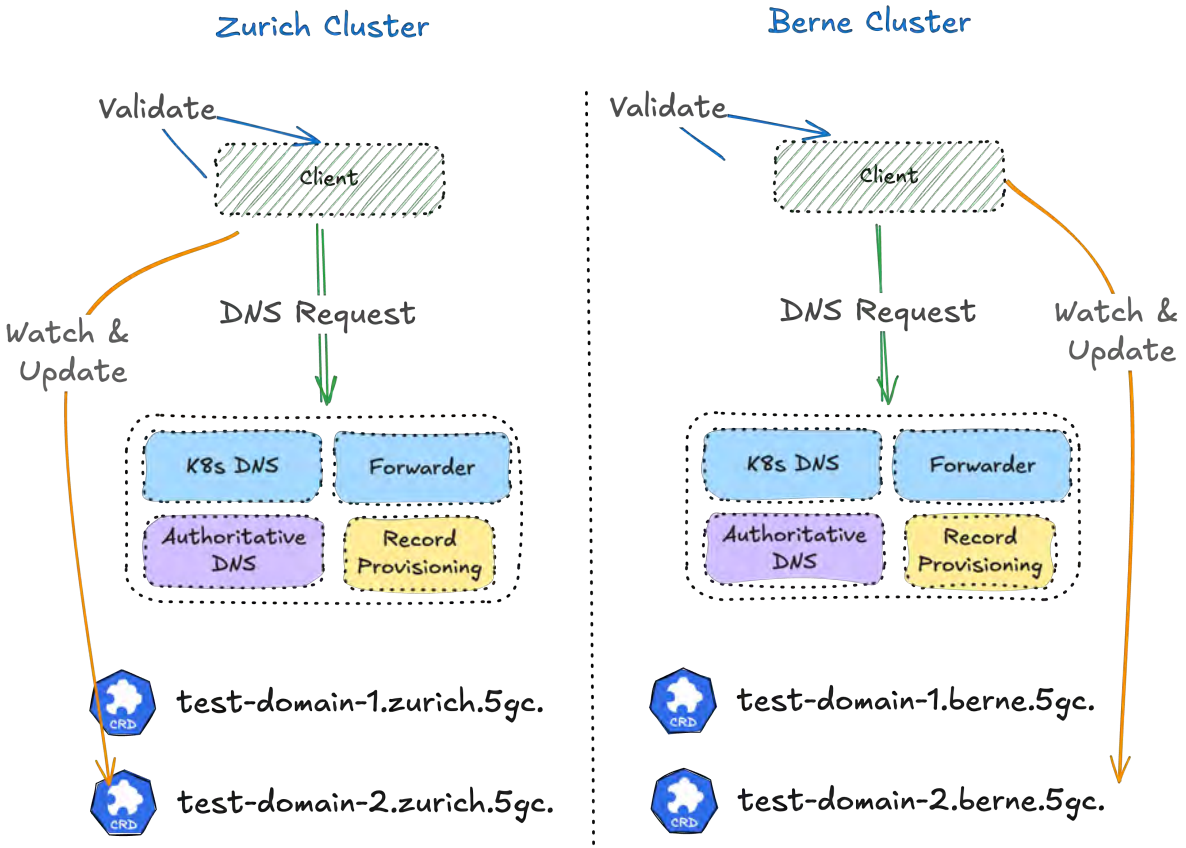
What is the source of truth?

- Authoritative Server?
- Authoritative Server Database?
- Quorum of Auth Servers?
- Cluster K8s API State?

Challenge of temporal coupling

⚠ More checks, more uncertainty.

How → DNS Client -- Traffic Patterns



Client becomes the source of truth.

- Maintains cluster state internally
- Issues updates
- Compares resolution against internal state
- Minimised temporal coupling

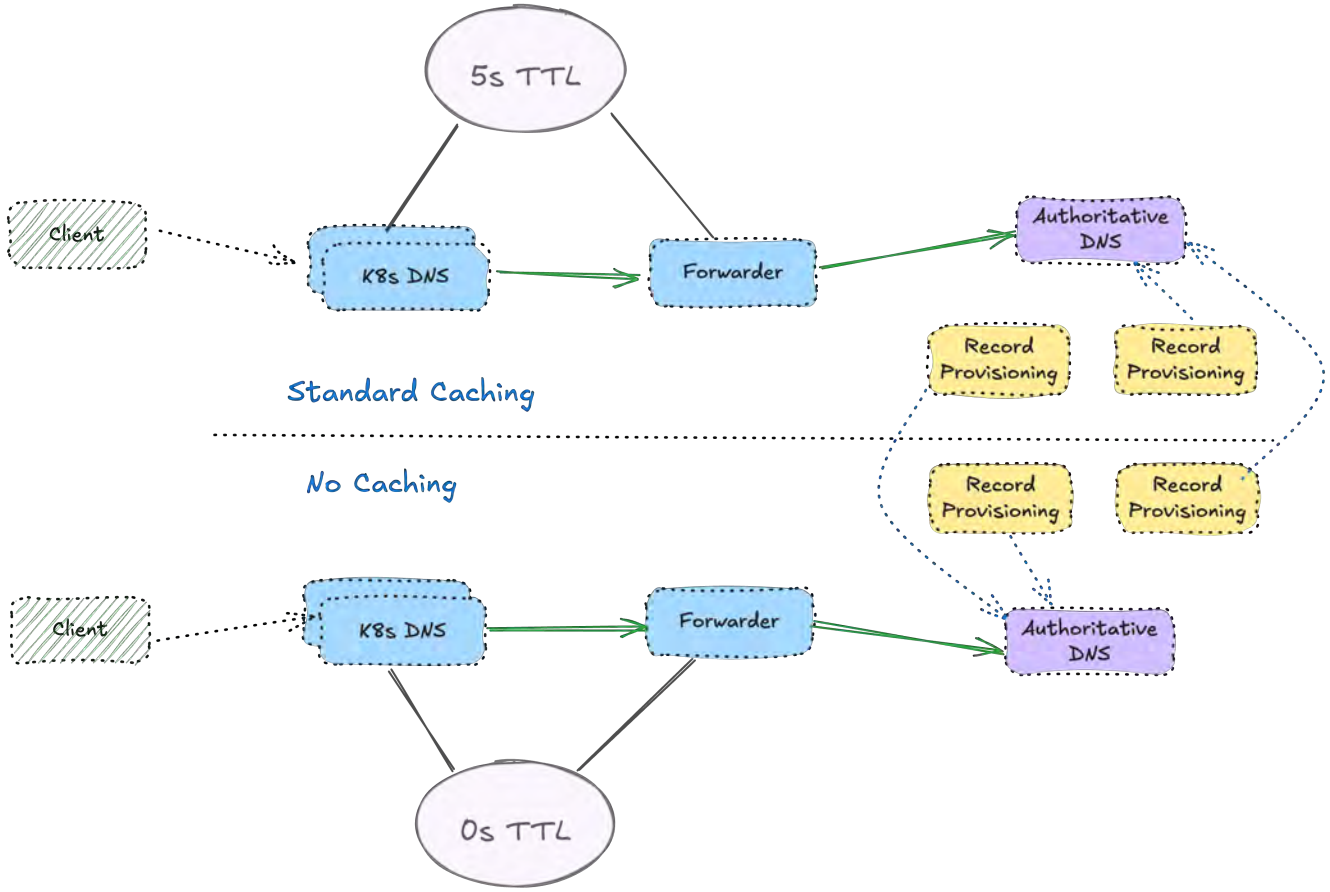
Traffic Patterns:

- Disjoined domain sets in each cluster
- Client only validates records in its cluster

Limitation over cross cluster validation.

Implementation based on Hostlookuper

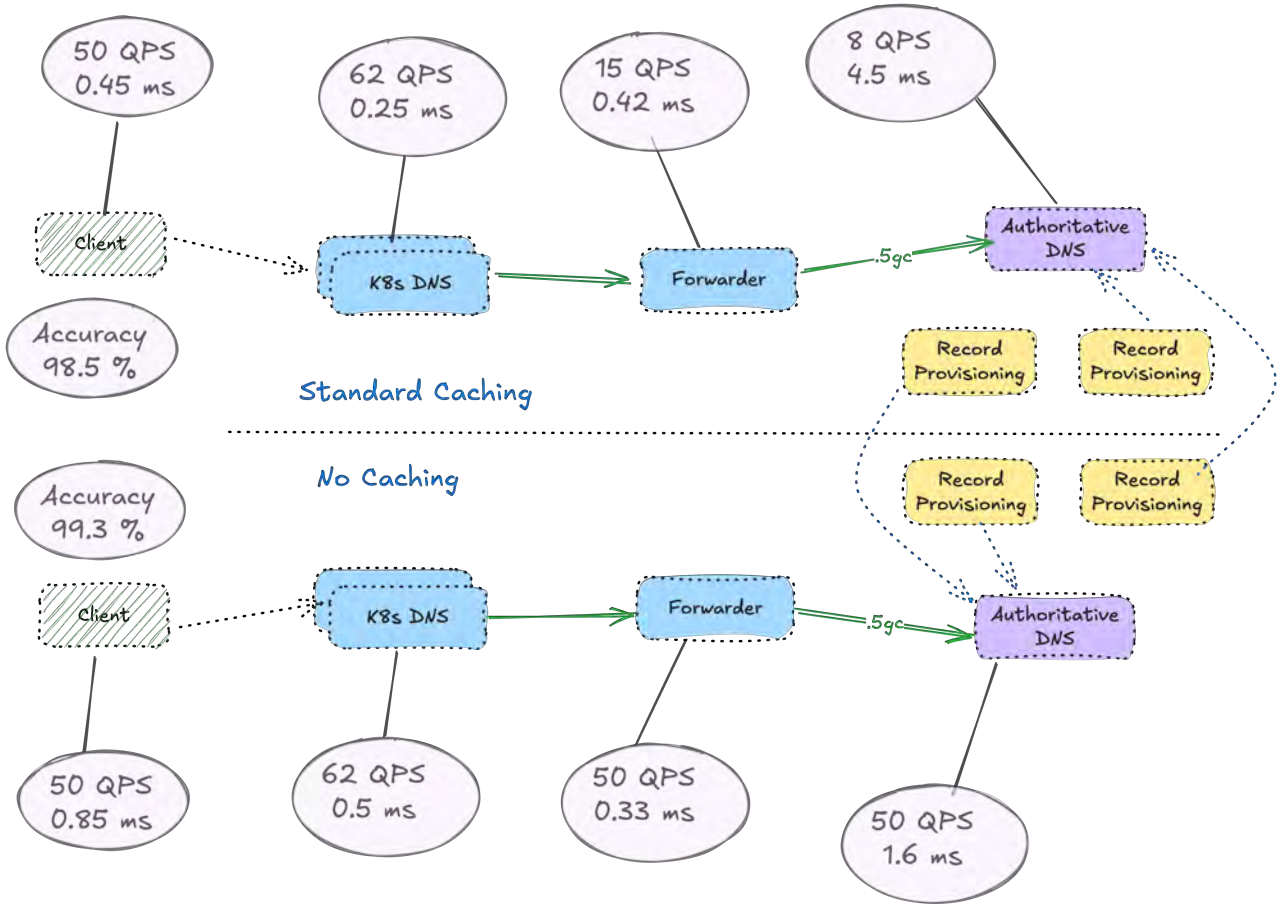
How → Caching Configurations -- Steady State



2 Caching Configurations:

- 5s TTL – Standard Caching
- No Caching

How → Caching Configurations -- Steady State



Steady State Measurements:

- One Client per Cluster
- 50 Unique Domains per Second

Resilience Testing on Cloud Native DNS

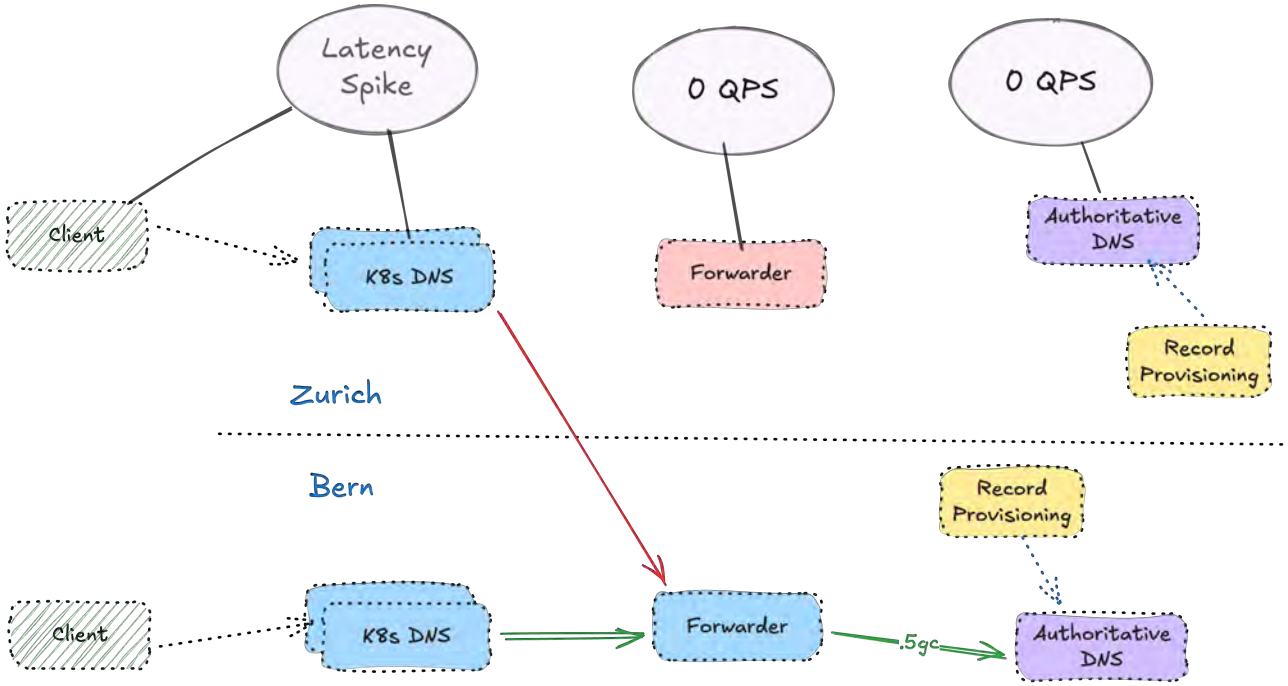
How → Chainsaw -- Datadog Chaos Injector

```
=== RUN   chainsaw/k8s-coredns-chaos-test
    | 14:46:03 | k8s-coredns-chaos-test | @chainsaw                | CREATE   | OK    |
    | 14:46:03 | k8s-coredns-chaos-test | Post Test Start          | TRY      | BEGIN |
    | 14:46:03 | k8s-coredns-chaos-test | Post Test Start          | CMD      | RUN   |
      === COMMAND
      /usr/bin/sh -c ../utils/annotate-grafana.sh test-k8s-dns-start
    | 14:46:03 | k8s-coredns-chaos-test | Post Test Start          | SCRIPT   | LOG   |
      === STDOUT
      {"id":124,"message":"Annotation added"}
    | 14:46:03 | k8s-coredns-chaos-test | Post Test Start          | SCRIPT   | DONE  |
    | 14:46:03 | k8s-coredns-chaos-test | Post Test Start          | TRY      | END   |
    | 14:46:03 | k8s-coredns-chaos-test | Inject Chaos             | TRY      | BEGIN |
    | 14:46:03 | k8s-coredns-chaos-test | Inject Chaos             | APPLY    | RUN   |
    | 14:46:03 | k8s-coredns-chaos-test | Inject Chaos             | CREATE   | OK    |
    | 14:46:03 | k8s-coredns-chaos-test | Inject Chaos             | APPLY    | DONE  |
    | 14:46:03 | k8s-coredns-chaos-test | Inject Chaos             | ASSERT   | RUN   |
    | 14:48:04 | k8s-coredns-chaos-test | Inject Chaos             | ASSERT   | DONE  |
    | 14:48:04 | k8s-coredns-chaos-test | Inject Chaos             | TRY      | END   |
    | 14:48:04 | k8s-coredns-chaos-test | Post Test Stop           | TRY      | BEGIN |
    | 14:48:04 | k8s-coredns-chaos-test | Post Test Stop           | CMD      | RUN   |
      === COMMAND
      /usr/bin/sh -c ../utils/annotate-grafana.sh test-k8s-dns-end
    | 14:48:04 | k8s-coredns-chaos-test | Post Test Stop           | SCRIPT   | LOG   |
      === STDOUT
      {"id":125,"message":"Annotation added"}
    | 14:48:04 | k8s-coredns-chaos-test | Post Test Stop           | SCRIPT   | DONE  |
    | 14:48:04 | k8s-coredns-chaos-test | Post Test Stop           | TRY      | END   |
    | 14:48:04 | k8s-coredns-chaos-test | Validate Revival         | TRY      | BEGIN |
    | 14:48:04 | k8s-coredns-chaos-test | Validate Revival         | ASSERT   | RUN   |
    | 14:49:13 | k8s-coredns-chaos-test | Validate Revival         | ASSERT   | DONE  |
    | 14:49:13 | k8s-coredns-chaos-test | Validate Revival         | TRY      | END   |
    | 14:49:13 | k8s-coredns-chaos-test | Annotate System Restored | TRY      | BEGIN |
    | 14:49:13 | k8s-coredns-chaos-test | Annotate System Restored | CMD      | RUN   |
      === COMMAND
      /usr/bin/sh -c ../utils/annotate-grafana.sh test-k8s-dns-system-restored
    | 14:49:13 | k8s-coredns-chaos-test | Annotate System Restored | SCRIPT   | LOG   |
      === STDOUT
      {"id":126,"message":"Annotation added"}
    | 14:49:13 | k8s-coredns-chaos-test | Annotate System Restored | SCRIPT   | DONE  |
    | 14:49:13 | k8s-coredns-chaos-test | Annotate System Restored | TRY      | END   |
    | 14:49:13 | k8s-coredns-chaos-test | Inject Chaos             | CLEANUP  | BEGIN |
    | 14:49:13 | k8s-coredns-chaos-test | Inject Chaos             | DELETE   | OK    |
    | 14:49:44 | k8s-coredns-chaos-test | Inject Chaos             | CLEANUP  | END   |
    | 14:49:44 | k8s-coredns-chaos-test | @chainsaw                | CLEANUP  | BEGIN |
    | 14:49:44 | k8s-coredns-chaos-test | @chainsaw                | DELETE   | OK    |
    | 14:49:49 | k8s-coredns-chaos-test | @chainsaw                | CLEANUP  | END   |
--- PASS: chainsaw (226.74s)
    --- PASS: chainsaw/k8s-coredns-chaos-test (226.74s)
PASS
```

Test Steps via Chainsaw:

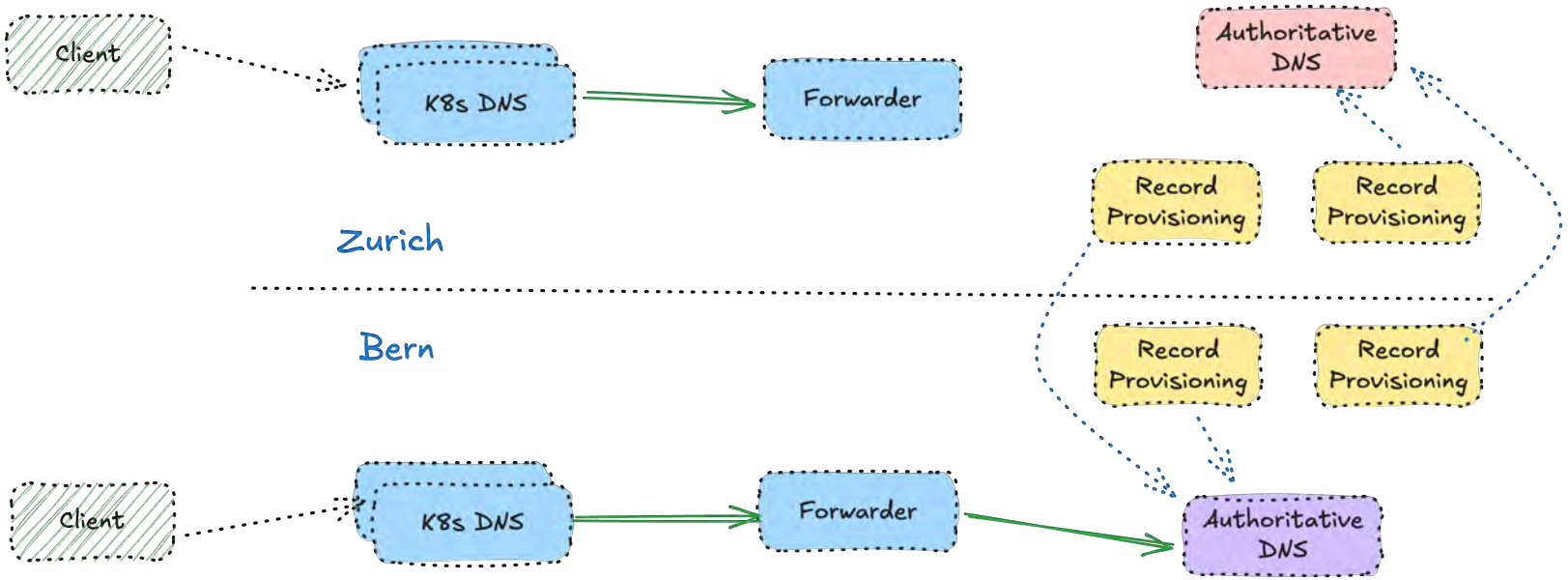
1. Annotate chaos injection start
2. Inject Chaos for 2 minutes
3. Annotate chaos injection finish
4. Wait for the component to recover
5. Annotate recovery

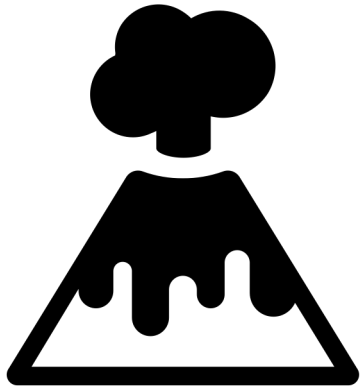
Evaluation → TC2 -- Forwarder Disruption



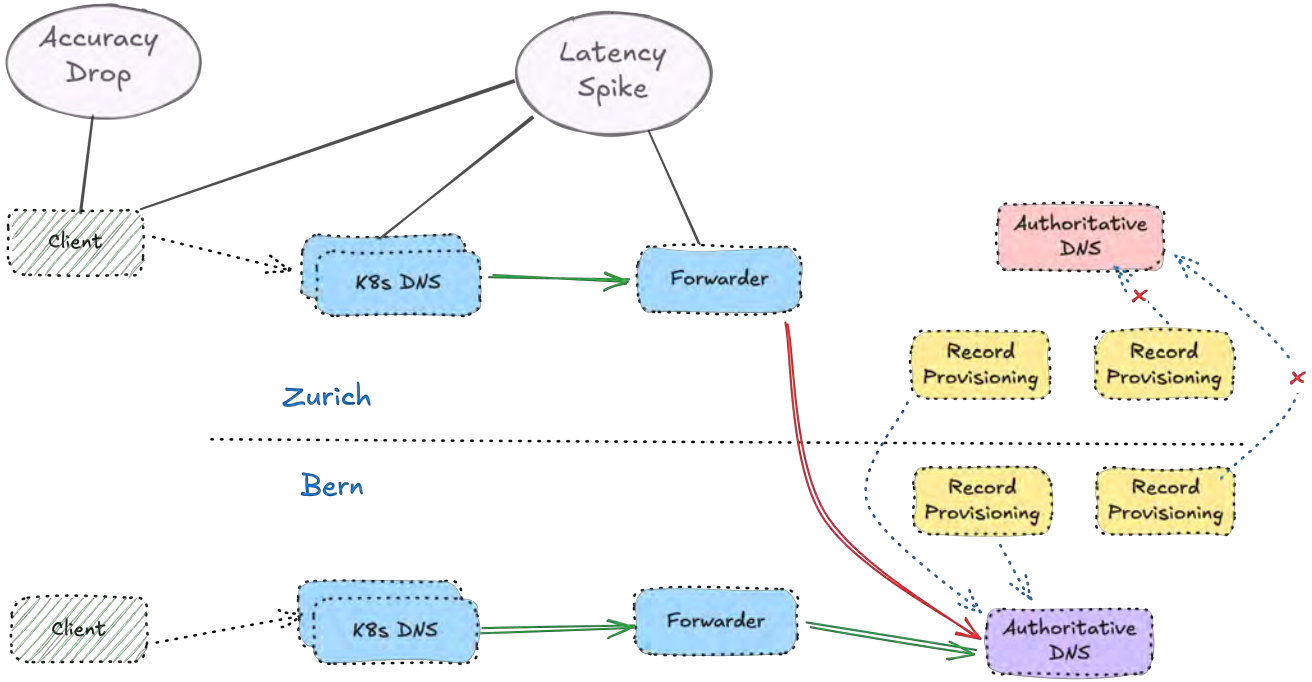
- Fallback route is used
- Latency spike in the beginning
- Higher latency spike for no-caching

Demo → TC3, Authoritative DNS Disruption





Evaluation → TC3 -- Authoritative DNS Disruption



Fallback route is used

Latency spike in the beginning

Higher latency spike for no-caching

Accuracy dips during restoration

Evaluation → Automation & Resilience

Fully Automated - GitOps driven DNS record provisioning

✓ Updates Propagate without manual intervention

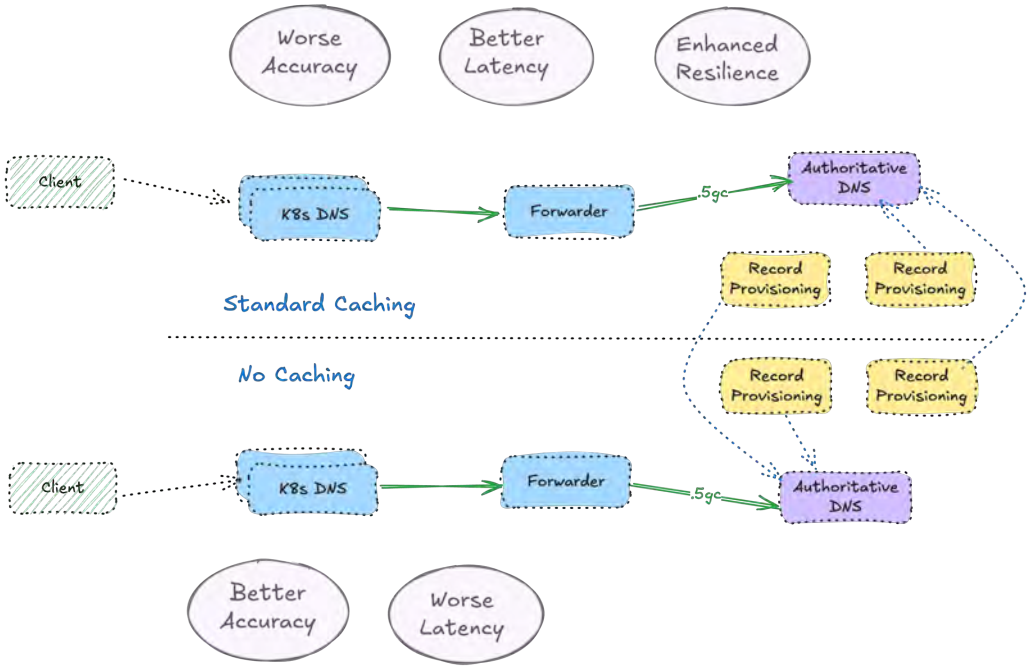
Geo-Redundant & HA
System remains operational under all tested disruptions

✓ Operational under disruption

Minimal Amount of SPOFs
Remove single points of failure from the System, Recover from Failure

✓ Cross-Cluster Forwarding
✓ Predictable Recovery

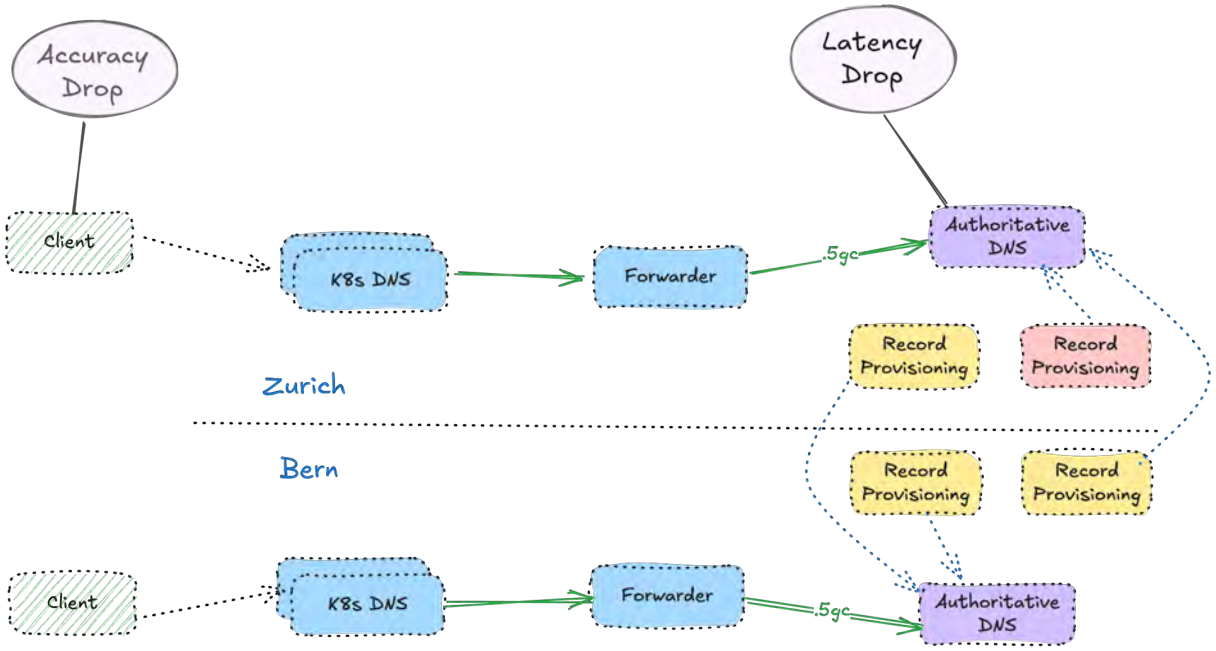
Evaluation → Caching



Caching improves resilience,
especially for failures occurred far from the client

No-caching, higher accuracy, worse latency
With TTL of 5s, Accuracy drop is estimated to ~0.5%

Evaluation → Testing Strategy



Testing strategy expose weaknesses

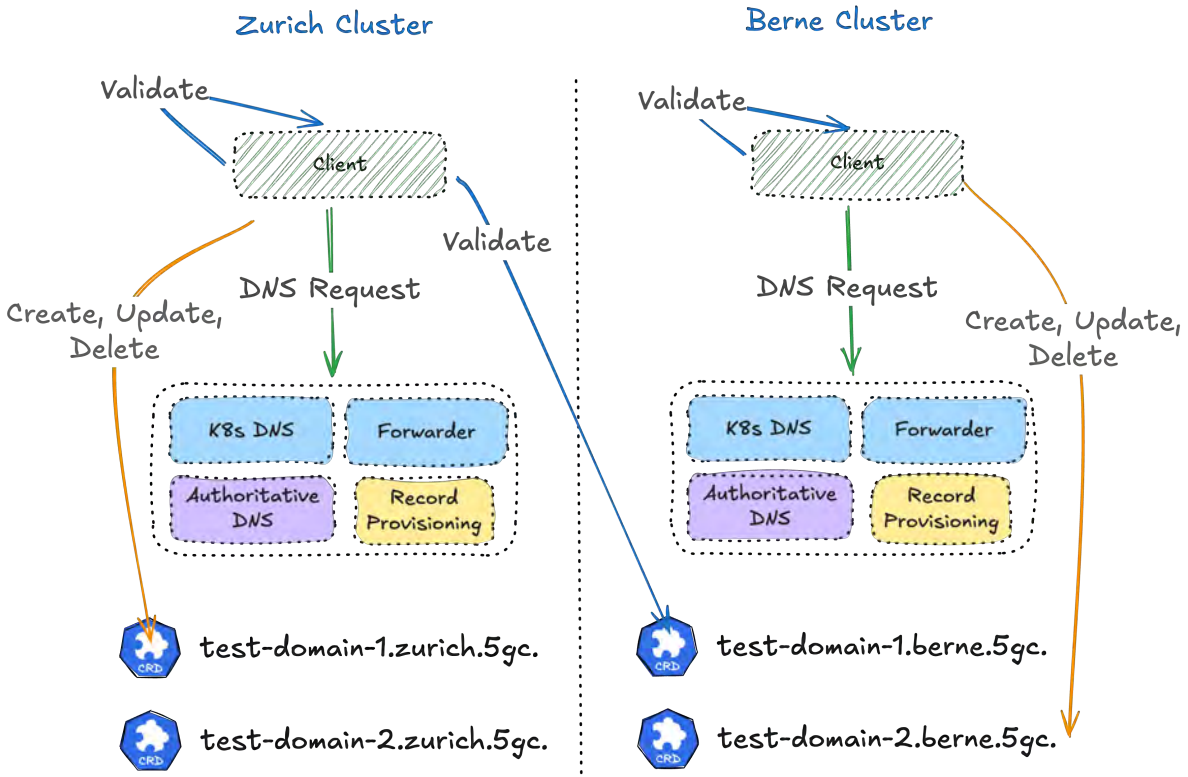
Client-side metrics →
validate resilience

Internal system metrics →
explain behaviour and root causes

Methodology easily extendable:

- Additional Clusters
- Alternative Caching Configurations

Future Work → Client Improvements

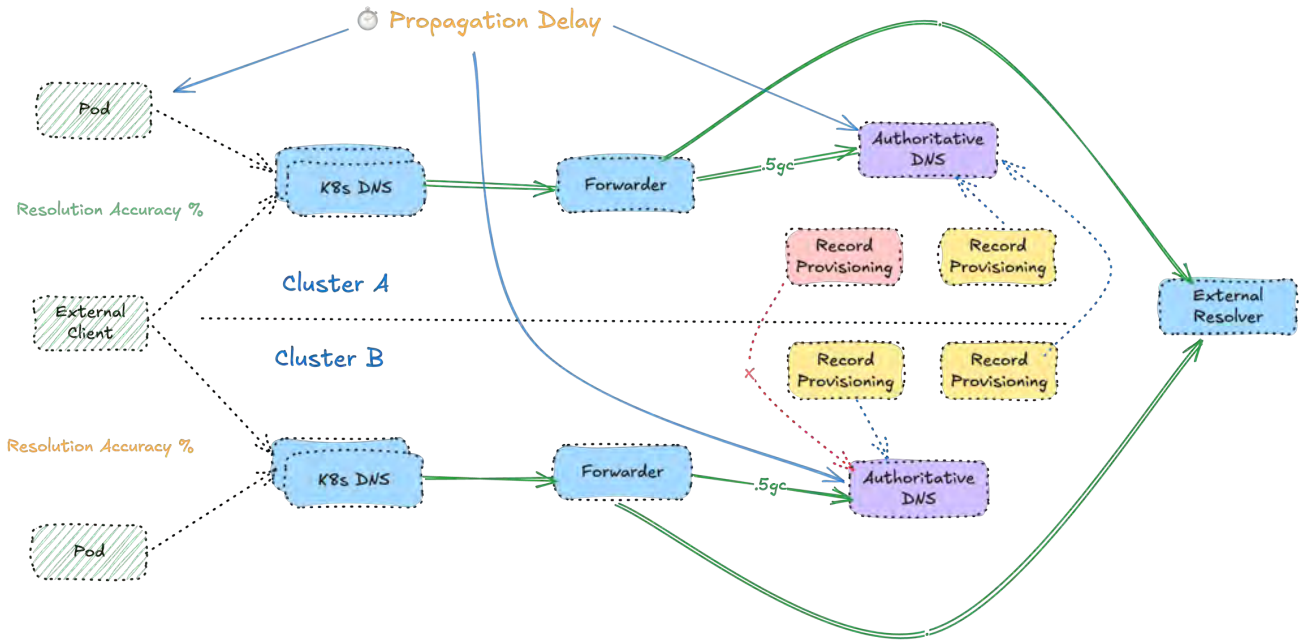


Limitations:
- Support only for updates
- Cross cluster validation

Improvements:
- Support Creation, Update & Deletion
- Cross Cluster lookups
- Fail fast delayed requests.

Resilience Testing on Cloud Native DNS

Future Work → Client Improvements



Estimate Propagation Delay

Measure time from record update to first correct client resolution

Error bounded by DNS query frequency

Thank you 🎉

Q&A