Virtual

CoreDNS for Hybrid and Multi-cloud

Yong Tang
Maintainer: CoreDNS
GitHub: yongtang

CoreDNS

CoreDNS

- Flexible DNS server written in Go
- Focus on service discovery
- Plugin based architecture, easily extended
- Memory safety (Golang) vs. BIND (C lang)
- Started and led by Miek Gieben

CoreDNS

CoreDNS: Cloud Native DNS

- Extended DNS Protocols
  - DNS over TLS
  - DNS over gRPC
- Default DNS server in Kubernetes
  - In Cluster
  - Out-of-Cluster
- Support Cloud Integration
  - AWS Route53
  - Google Cloud DNS
  - Azure DNS

CoreDNS

Corefile: CoreDNS Configurations


```
.:53 {
    # By default all plugins are disabled initially, unless enabled explicitly
    kubernetes cluster.local {                  # <- k8s integration
       pods insecure
    }
    route53 example.com.:Z1Z2Z3Z4DZ5Z6Z7        # <- route53 aws cloud data sync up
    hosts example.hosts example.org {           # <- additional records, added (inline)
       192.0.0.100 www.example.org
    }
    health                                      # <- healthcheck
    prometheus                                  # <- metrics
    cache 30                                    # <- cache & performance
    forward . 1.1.1.1:53                        # <- forward to 1.1.1.1 (Cloudflare)
    errors
}
```

CoreDNS

Plugins: Forward & Hosts


```
.:53 {
    # By default all plugins are disabled initially, unless enabled explicitly
    hosts example.hosts example.org {         # <- additional records, added (inline)
       192.0.0.100 www.example.org
    }
    forward . 8.8.8.8                         # <- forward to 8.8.8.8 (Google Public DNS)
    errors
}
```

$ dig @127.0.0.1 -p 53 www.example.org

……

;; ANSWER SECTION:

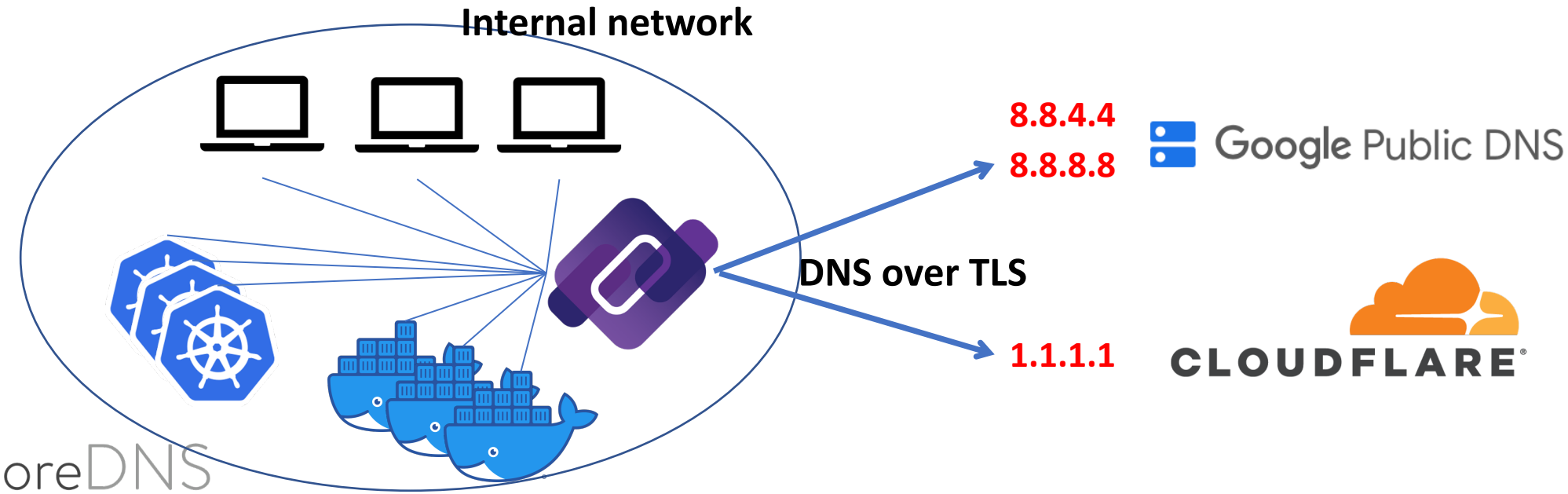
www.example.org. 3600 IN A 192.0.0.100

……

CoreDNS

DNS over TLS (1)

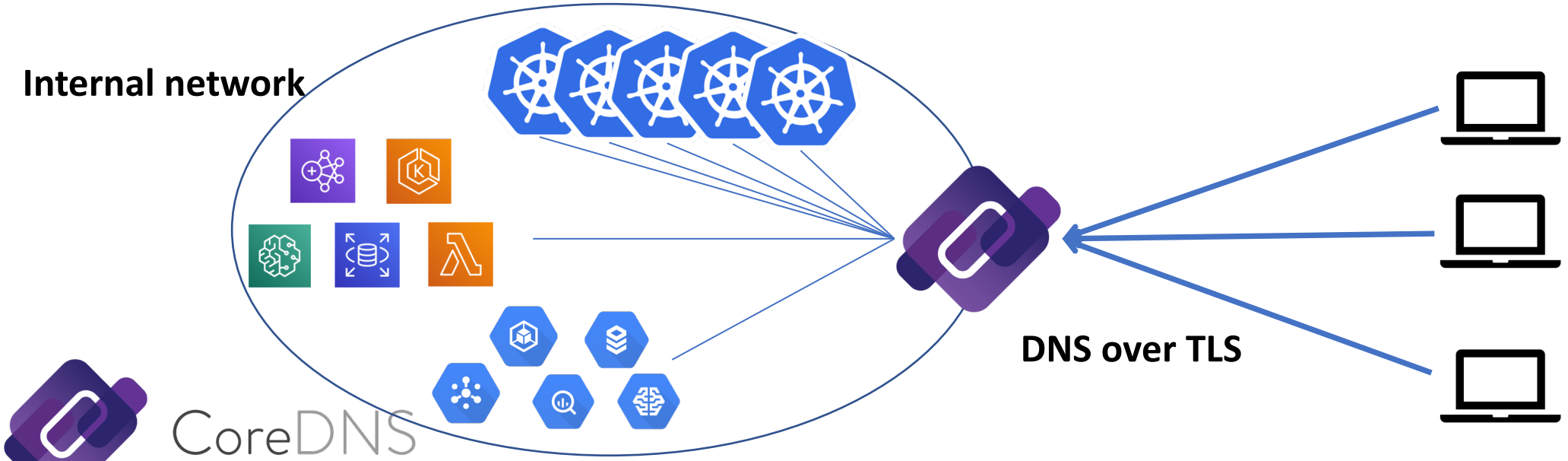
- Forward queries via TLS (Internet)
  - `.:53 {` `# <- serving DNS queries`
  - `    forward . tls://8.8.8.8 tls://8.8.4.4` `# <- forward to TLS backend`
  - `    debug`
  - `}`



CoreDNS

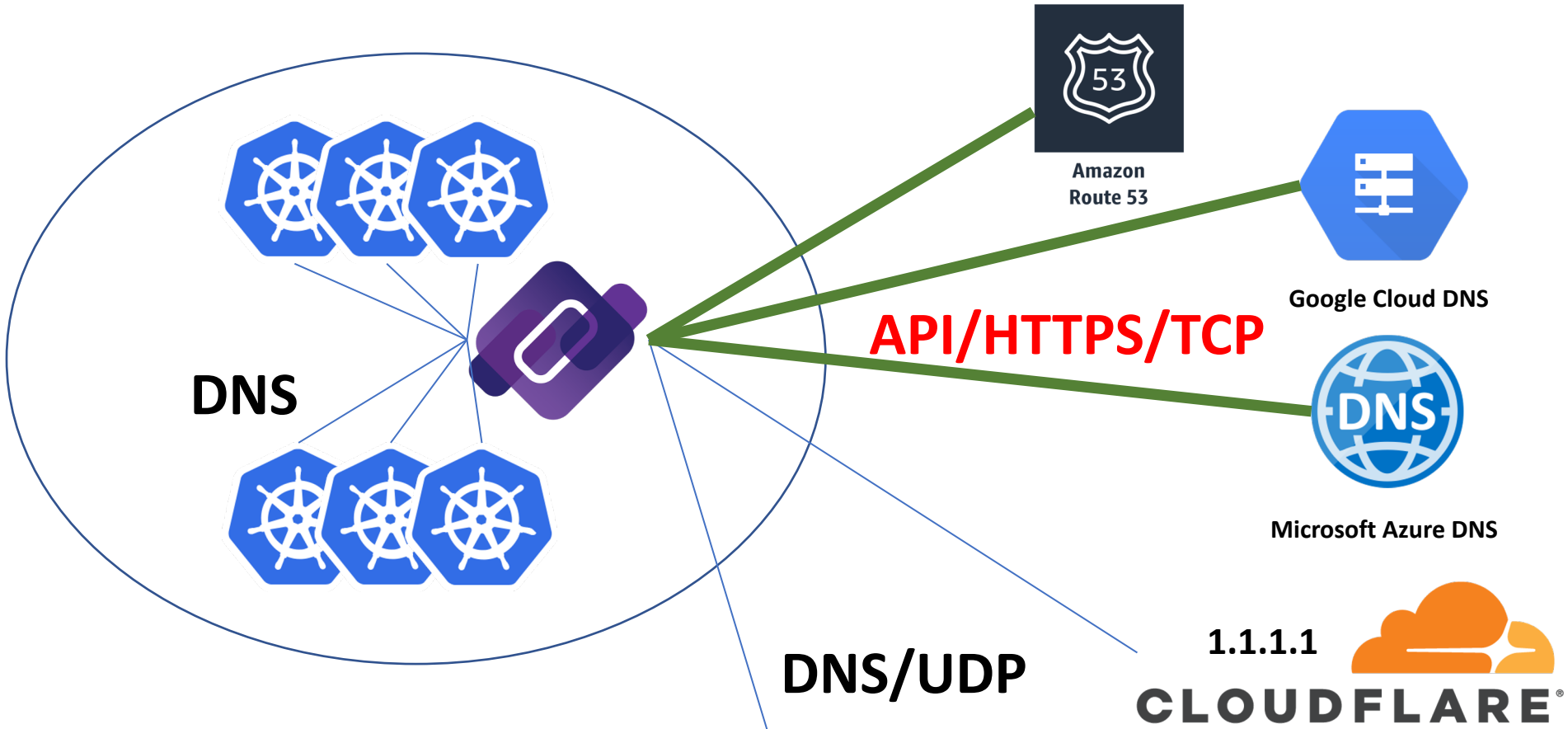
DNS over TLS (2)

- Serving DNS over TLS
  - `tls://.:53 {` `# <- serving queries over TLS`
  - `    tls MyCertificate.crt MyKey.key`
  - `    forward . 8.8.8.8 8.8.4.4` `# <- forward to UDP backend`
  - `    debug`
  - `}`



CoreDNS

Service Discovery




```
# /ets/hosts like inline records
hosts example.hosts example.org {
   192.0.0.100 www.example.org
}
```

CoreDNS

CoreDNS for Service Discovery

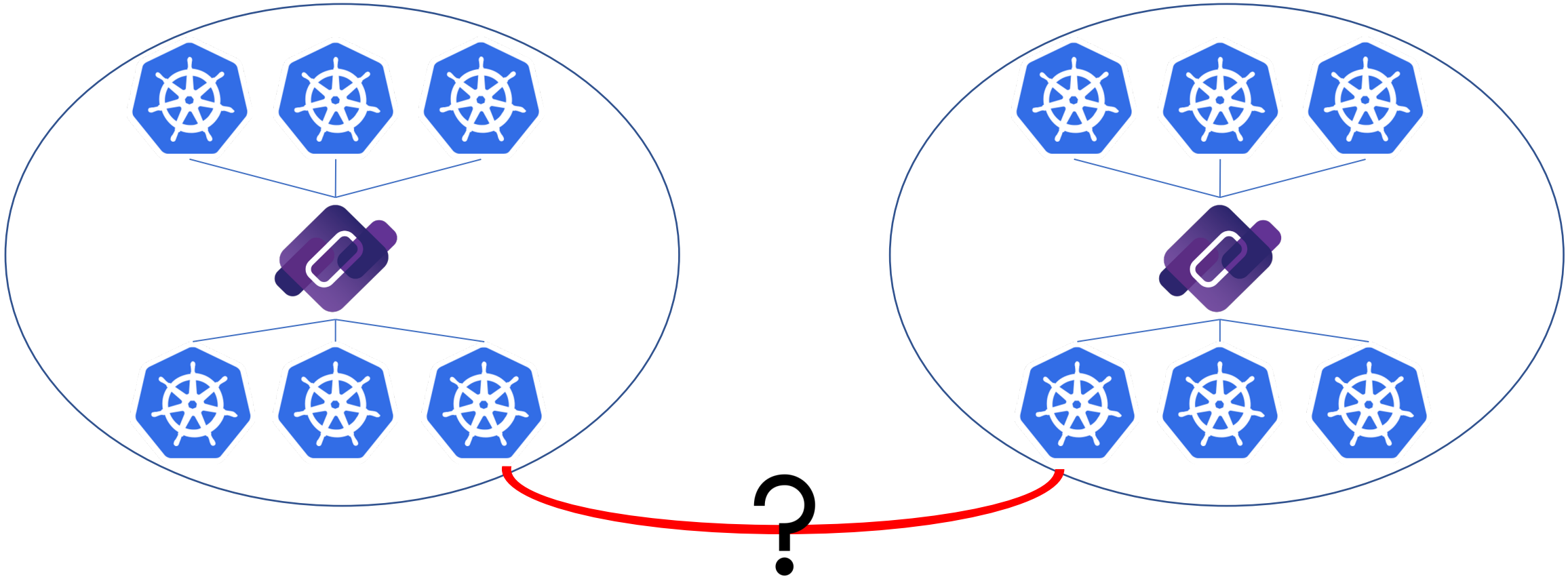
- DNS is a nice indirection
- Maximum flexibility
- Easy and simple to change
- Distributed in nature, scales to Internet
- DNS Pervasive in IT infrastructure

Data from Different Sources


```
.:53 {
    # By default all plugins are disabled initially, unless enabled explicitly
    kubernetes cluster.local {                # <- k8s integration
       pods insecure
    }
    route53 example.com.:Z1Z2Z3Z4DZ5Z6Z7      # <- route53 aws cloud data sync up
    hosts example.hosts example.org {.        # <- additional records, added (inline)
       192.0.0.100 www.example.org
    }
    health                                    # <- healthcheck
    prometheus                                # <- metrics
    cache 30                                  # <- cache & performance
    forward . 1.1.1.1:53                      # <- forward to 1.1.1.1 (Cloudflare)
    errors
}
```

CoreDNS

Service Discovery: Multi-Cluster



How to make pods inter-reachable across clusters

CoreDNS

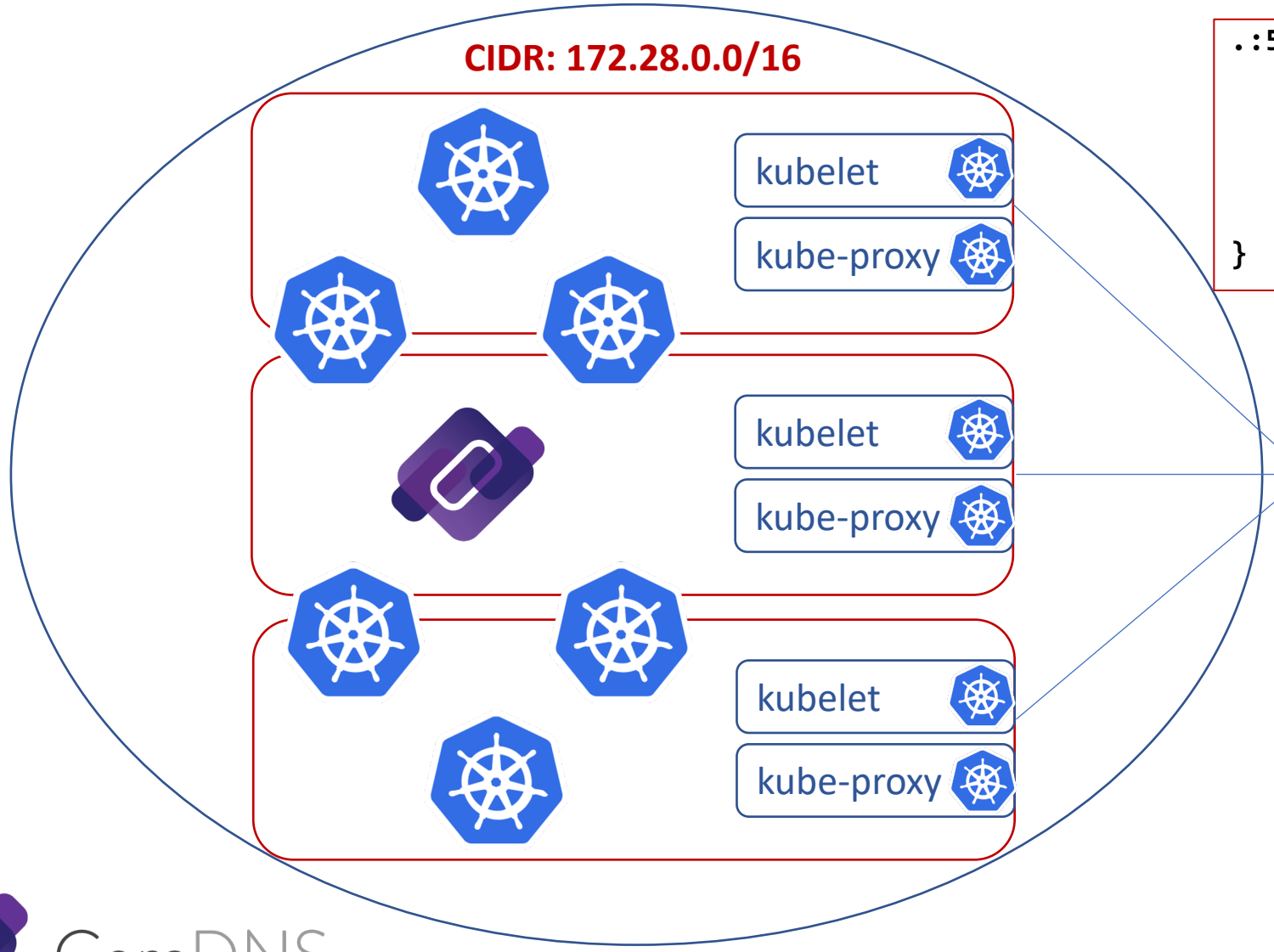
Service Discovery: Multi-Cluster

Prerequisite:

- Non-Overlapping cluster CIDR
- All IPs routable in any cluster
- Headless services only
- No ClusterIP services

CoreDNS

CoreDNS: In-Cluster Deploy



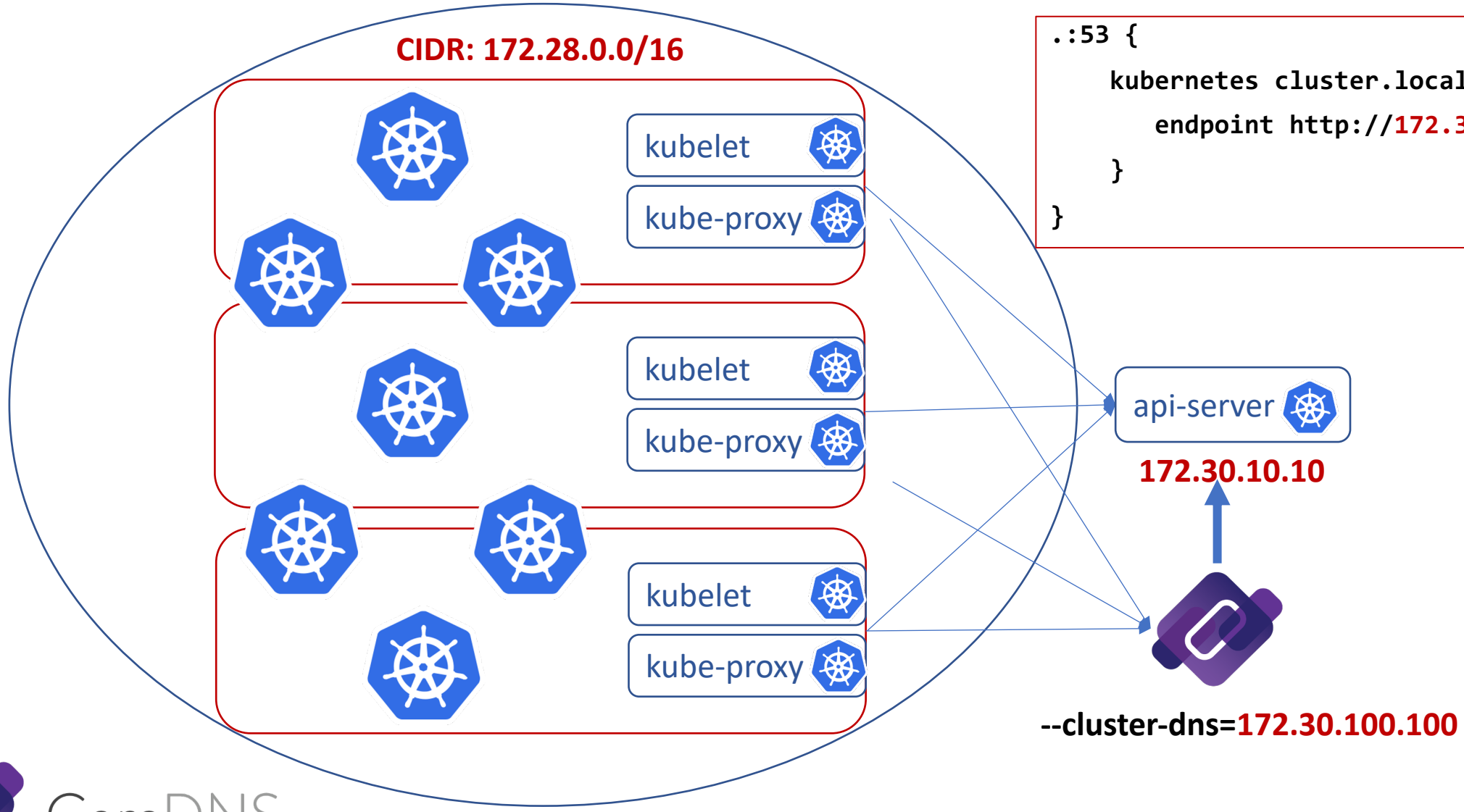

```
.:53 {
    kubernetes cluster.local {
       pods insecure
    }
}
```

api-server

172.30.10.10

CoreDNS

CoreDNS: Out-of-Cluster Deploy



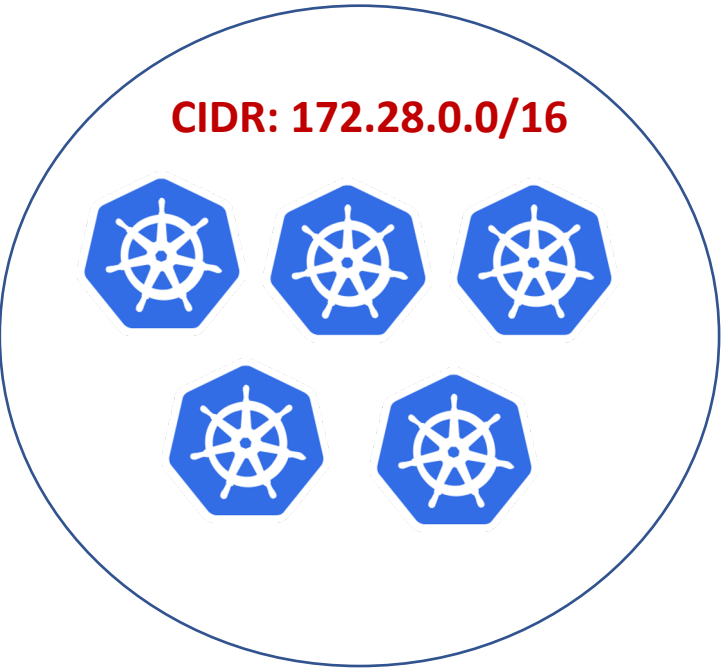
CoreDNS

Service Discovery: Multi-Cluster

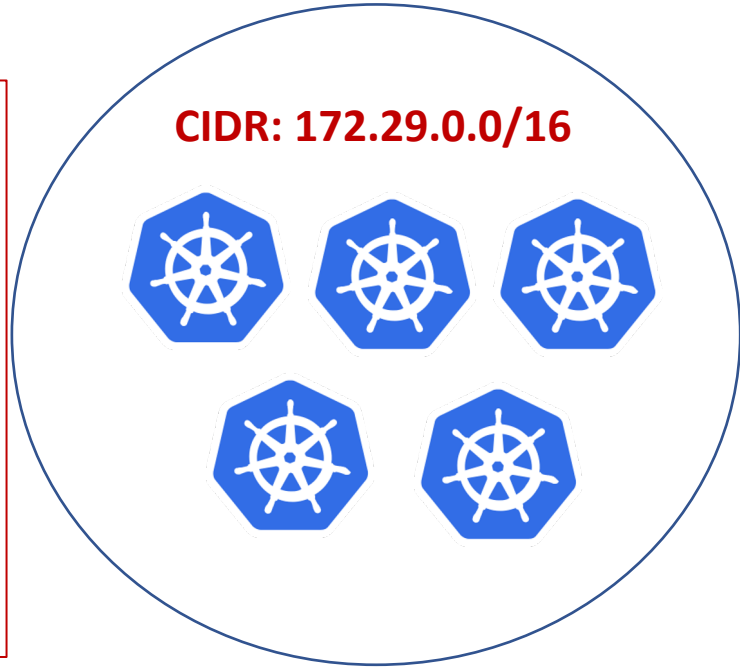
172.30.10.10

172.30.20.10

--cluster-dns=172.30.100.100



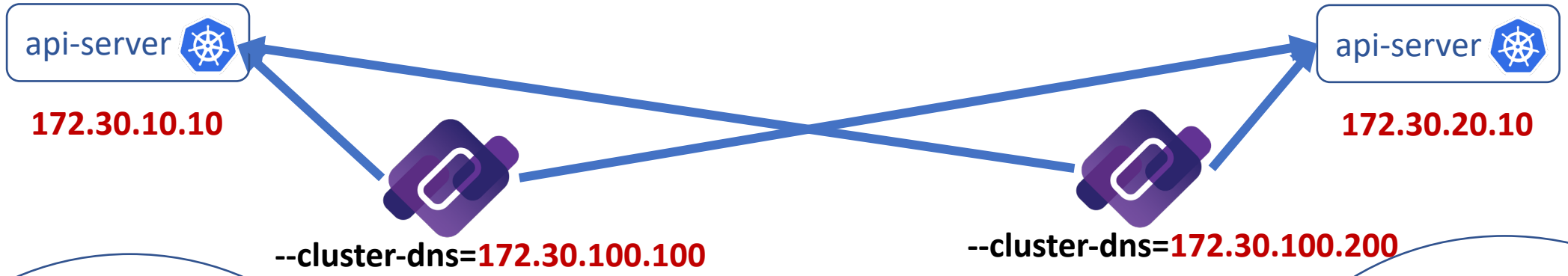

```
.:53 {
    kubernetai cluster1.local {
       endpoint http://172.30.10.10
    }
    kubernetai cluster2.local {
       endpoint http://172.30.10.20
    }
}
```



kubernetai plugin

CoreDNS

Fallthrough & Fault Tolerance



CIDR: 172.28.0.0/16

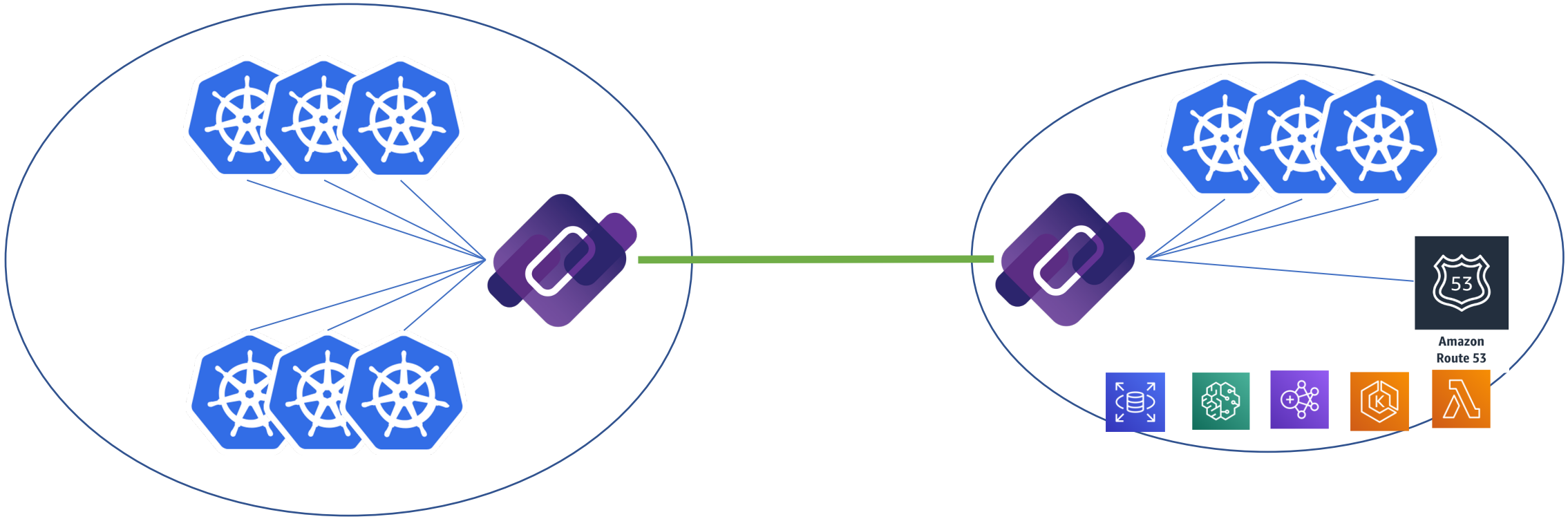
CIDR: 172.29.0.0/16

```
.:53 {
    kubernetai cluster.local {
       ignore empty_service
       fallthrough
    }
    kubernetai cluster.local {
       endpoint http://[172.30.10.10|172.30.20.10]
    }
}
```

kubernetai plugin

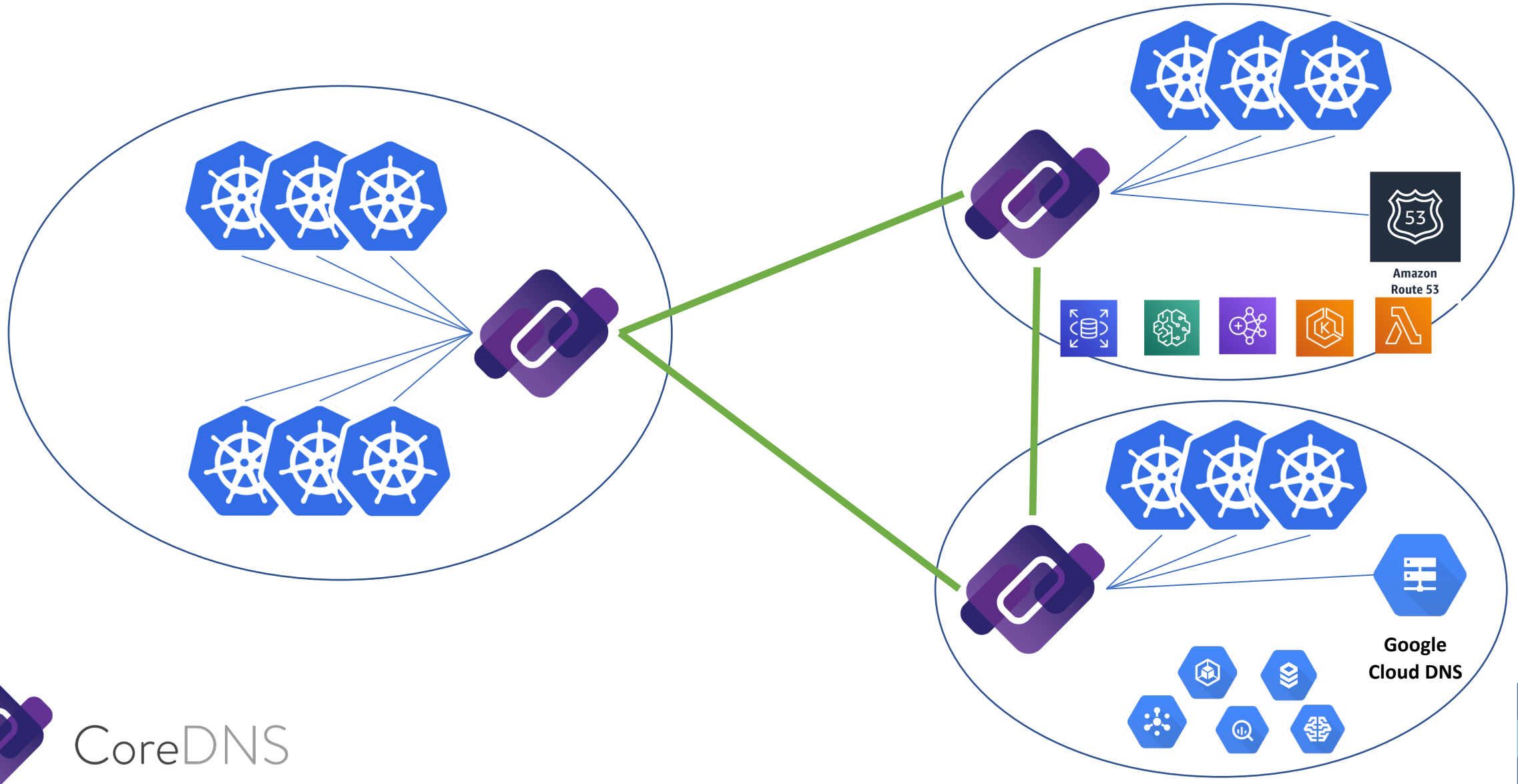
CoreDNS

Service Discovery: Hybrid-Cloud



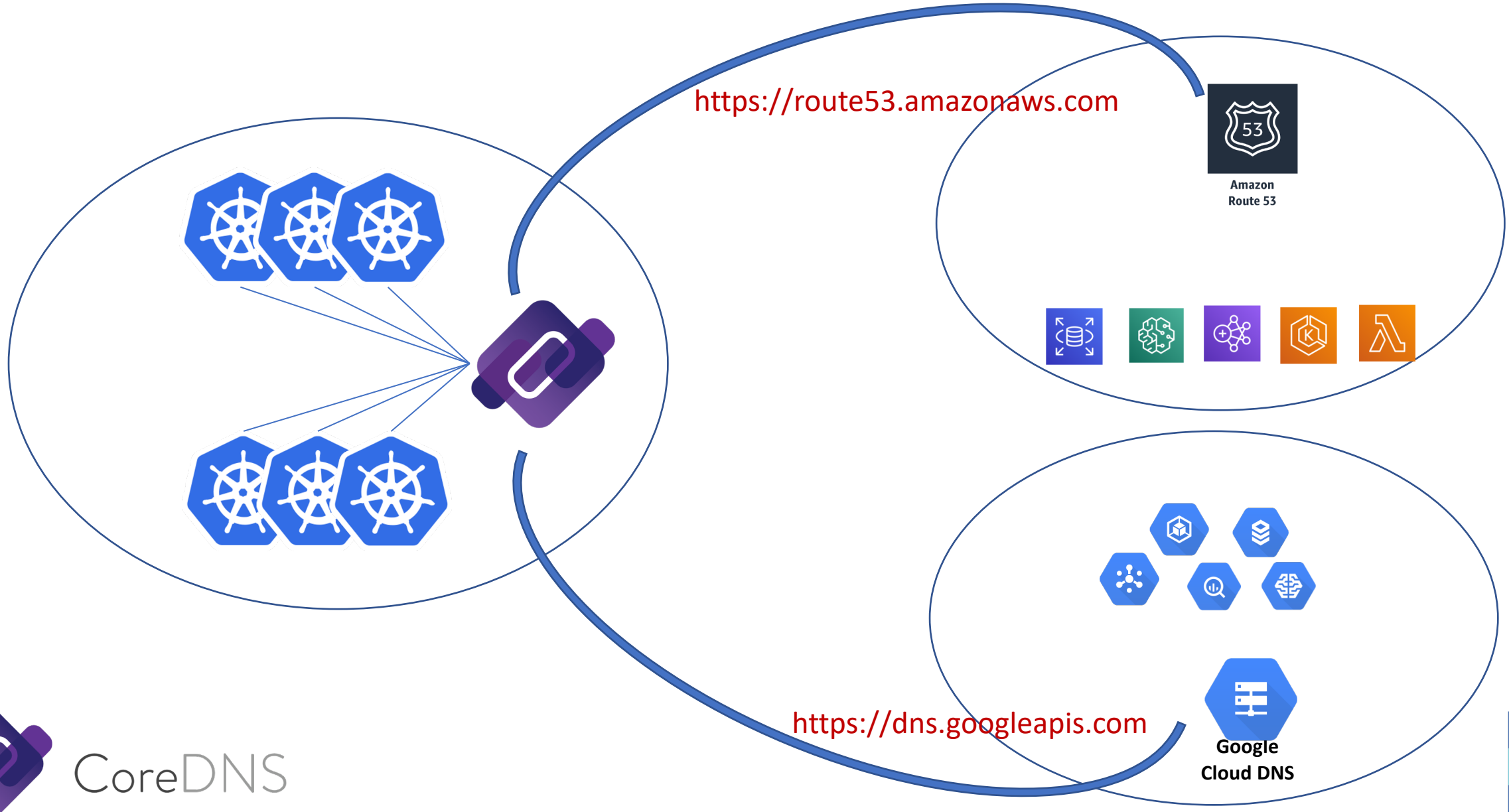
CoreDNS

Service Discovery: Multi-Cloud



CoreDNS

CoreDNS: Cloud Integration

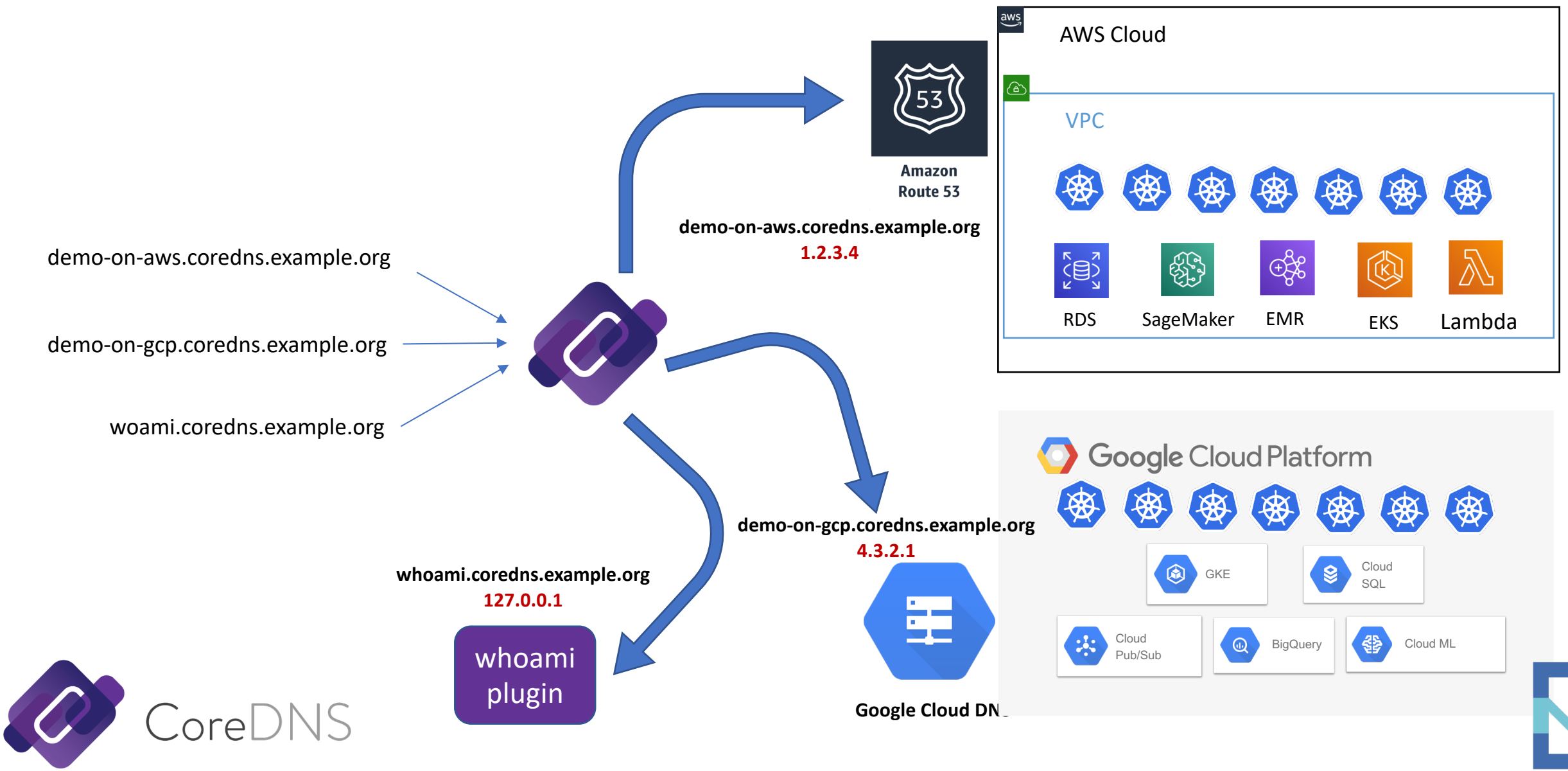


CoreDNS

CoreDNS: Cloud Integration

- Secure communication (HTTPS vs UDP)
- Authentication and authorization
- Reliable and better error handling (TCP vs UDP)
- Separation of data sync up & DNS query

CoreDNS: AWS + Google Cloud



CoreDNS: AWS + Google Cloud


```
.:53 {
    # Route53 (Amazon AWS)
    route53 coredns.example.org.:Z01234567890123456789 {
       fallthrough      # <- move to next plugin (clouddns)
    }
    # Cloud DNS (Google Cloud)
    clouddns coredns.example.org.:peerless-dahlia-123456:coredns-example-zone {
       fallthrough      # <- move to next plugin (e.g., whoami)
    }
    # Fallthrough (e.g., whoami)
    whoami
    # …
}
```

CoreDNS

THANK YOU

CoreDNS


KubeCon
CloudNativeCon
Europe 2020
Virtual
HELM
KEEP CLOUD NATIVE
CONNECTED