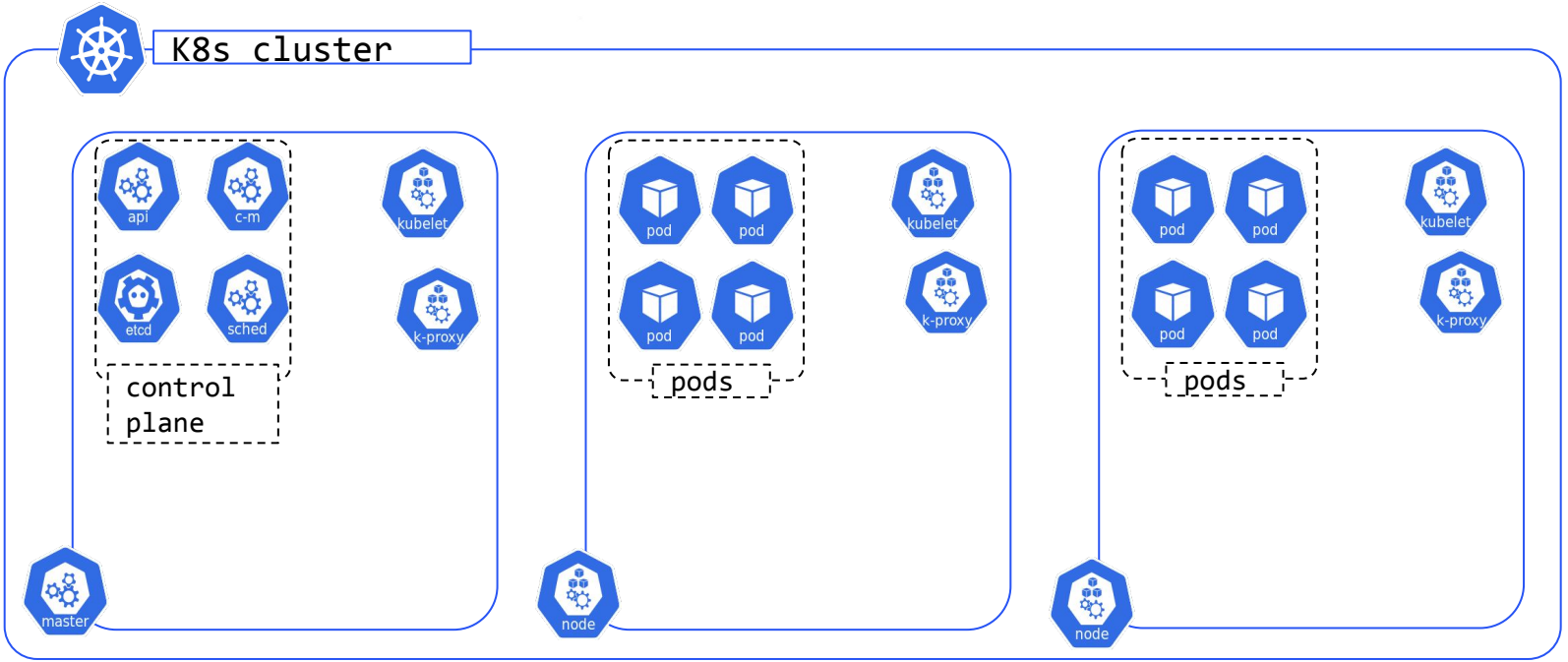

KubeCon
CloudNativeCon
Europe 2023
TiKV

Demystifying IPv6 Kubernetes

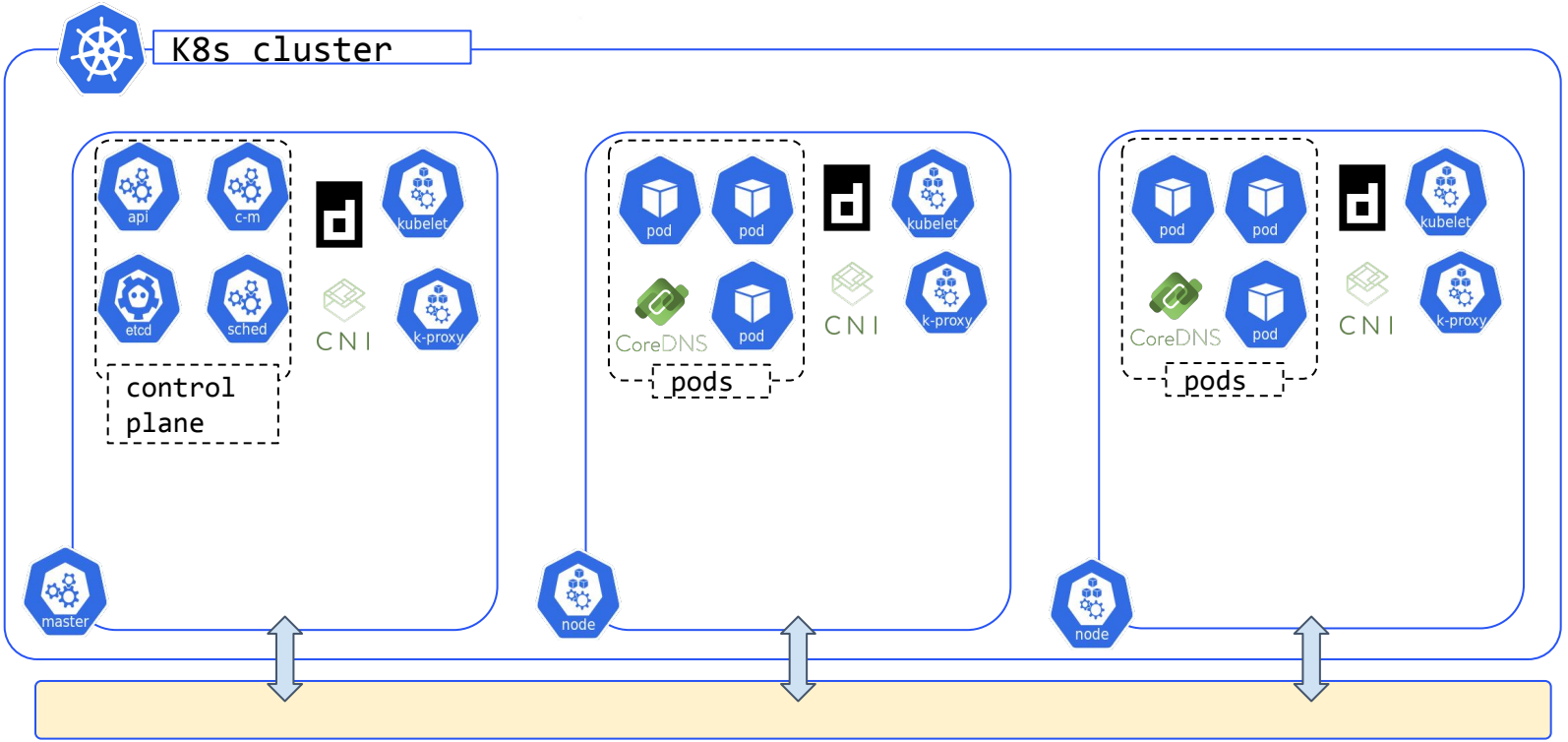

le coq sportif

Kubernetes Networking Architecture

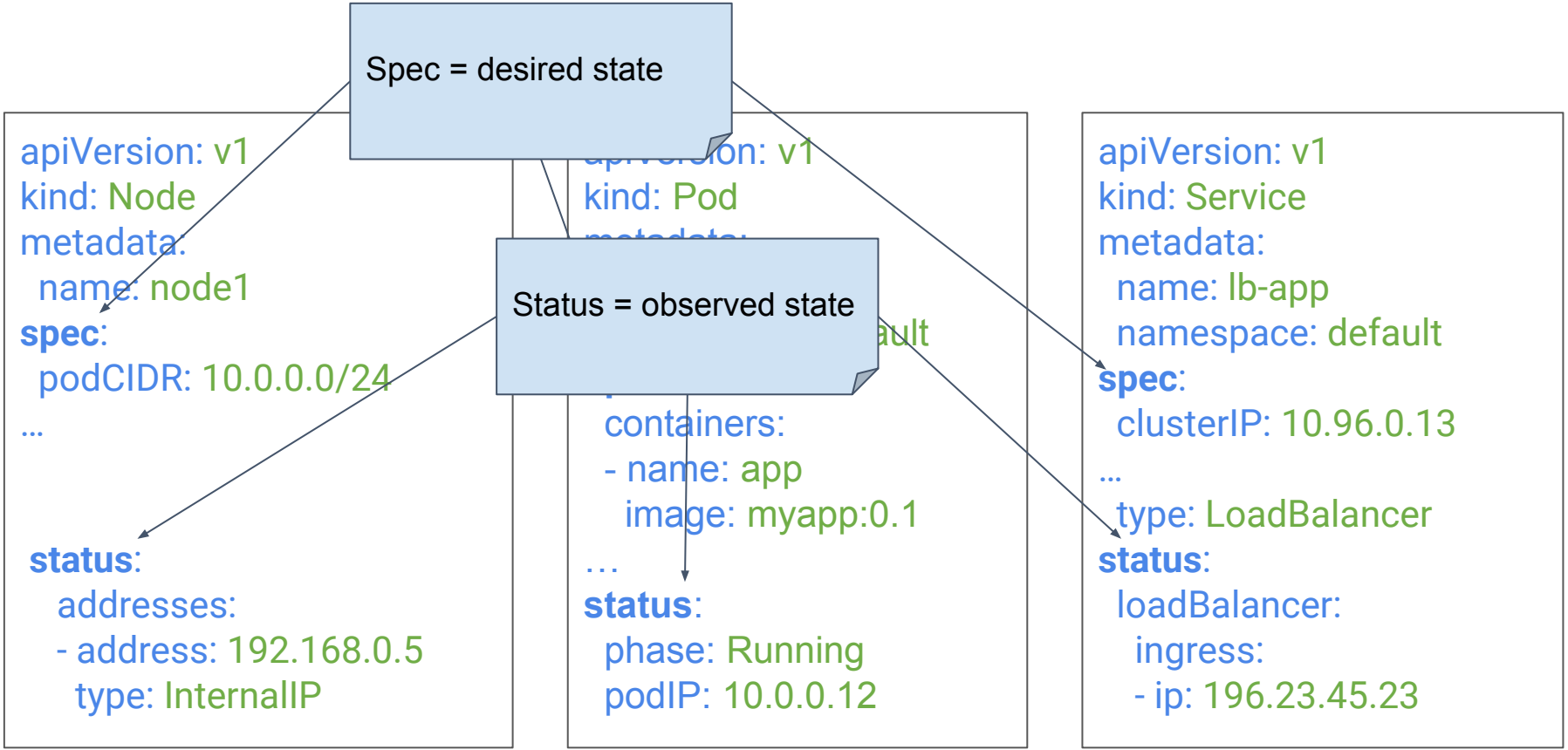
Kubernetes reference architecture



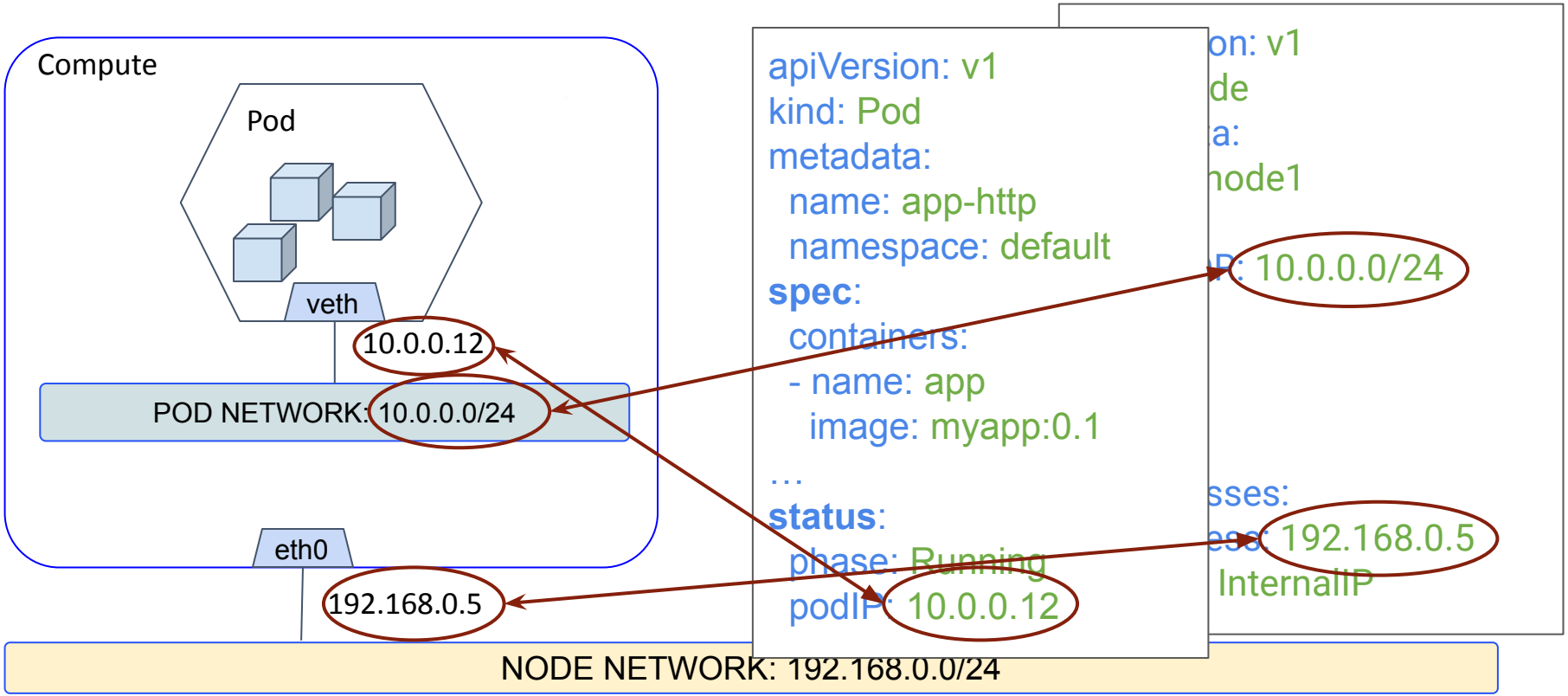
Kubernetes reference implementation



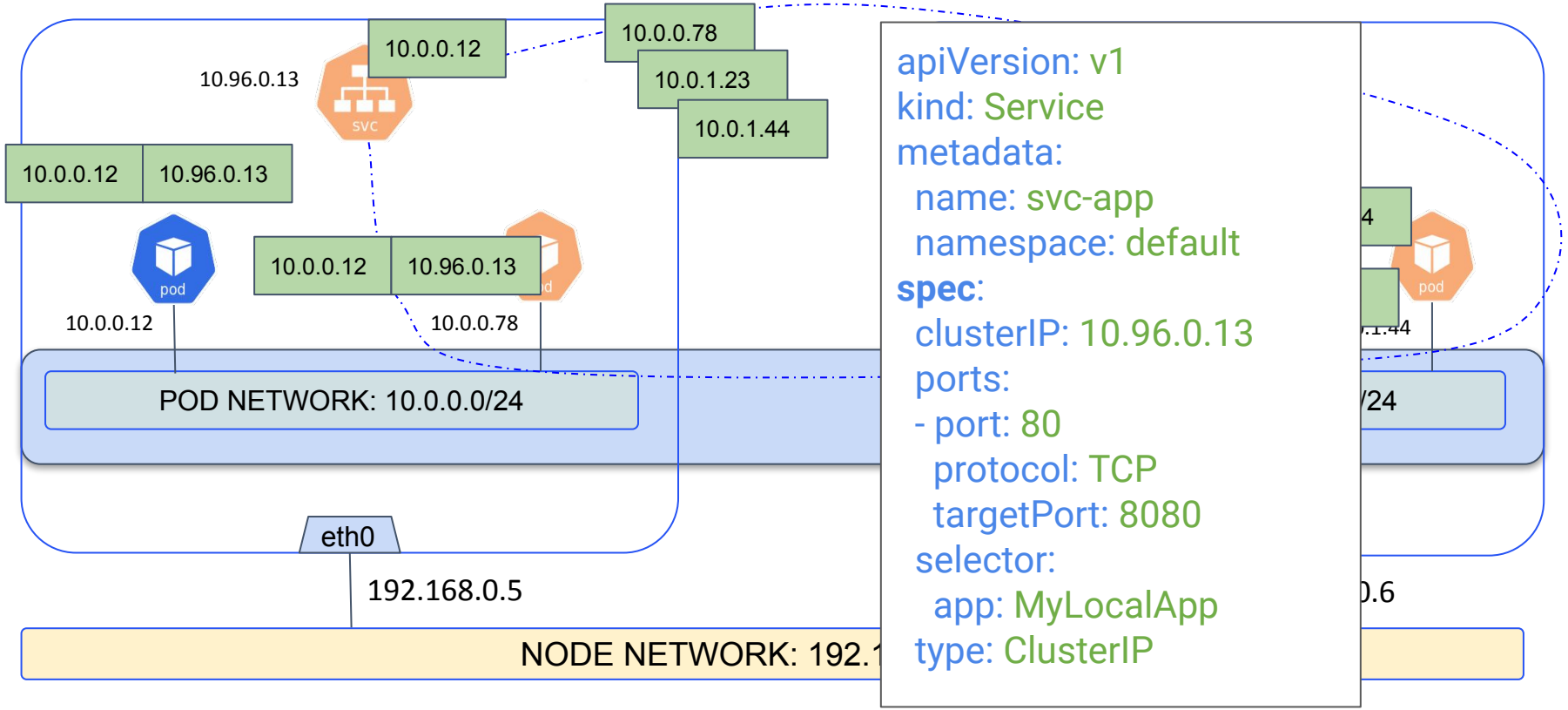
Kubernetes API



Kubernetes Networking: Pods

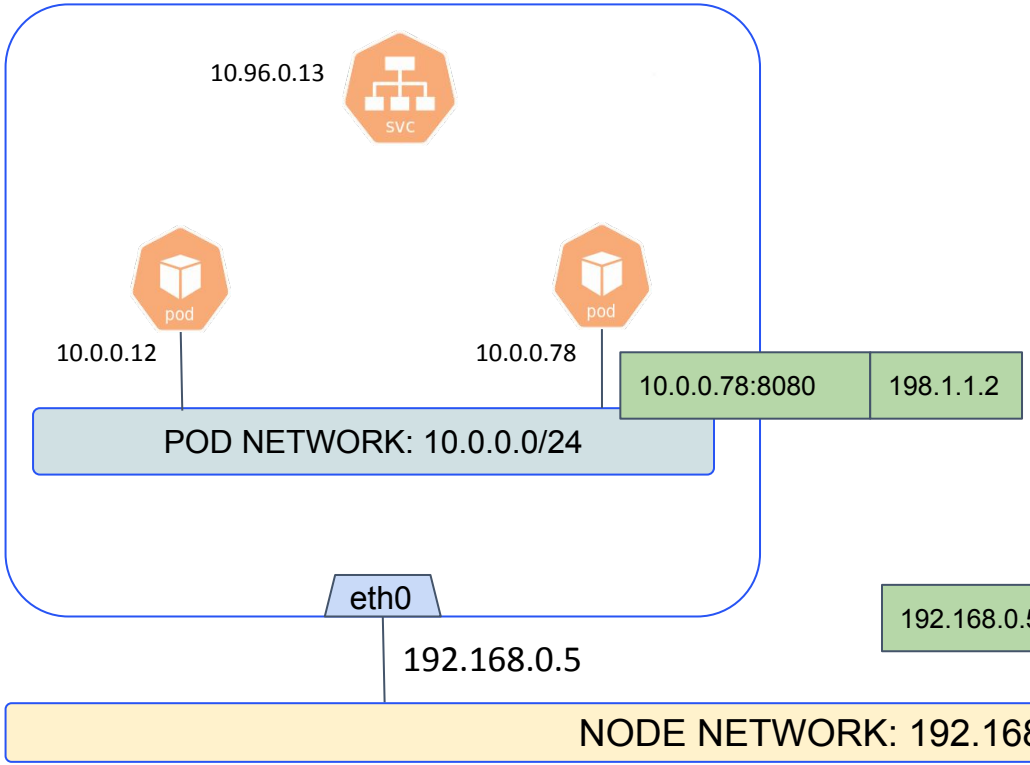


Kubernetes Networking: Services




```
apiVersion: v1
kind: Service
metadata:
  name: svc-app
  namespace: default
spec:
  clusterIP: 10.96.0.13
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: MyLocalApp
  type: ClusterIP
```

Kubernetes Networking: Services



196.

192.168.0.5:32101

```yaml
apiVersion: v1
kind: Service
metadata:
  name: lb-app
  namespace: default
spec:
  clusterIP: 10.96.0.13
  ports:
  - nodePort: 32101
    port: 80
    protocol: TCP
    targetPort: 8080
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 196.23.45.23
```

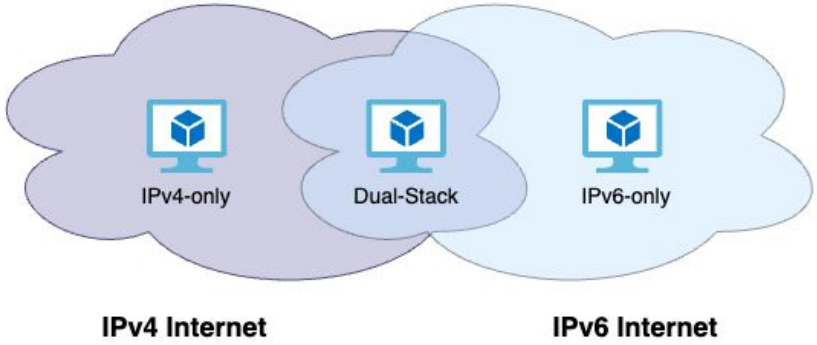
Internet Protocol version 6 (IPv6)

Brief history of IPv6

- The Internet relies on unique IP addresses for host communication
- But 20+ years ago it was evident IPv4 addresses would be exhausted
- Short-term solutions (stop-gaps):
  - Classless Internet Domain Routing (CIDR)
  - Variable Length Subnet Masking (VLSM)
  - NAT (industry-driven)
- Long term solution:
  - IPv6

IPv6 in a Nutshell

- Designed to tackle the problem of IPv4 address exhaustion
- Backwards **incompatible** with IPv4
- Original transition/deployment model: dual-stack



IPv6 Addressing Overview

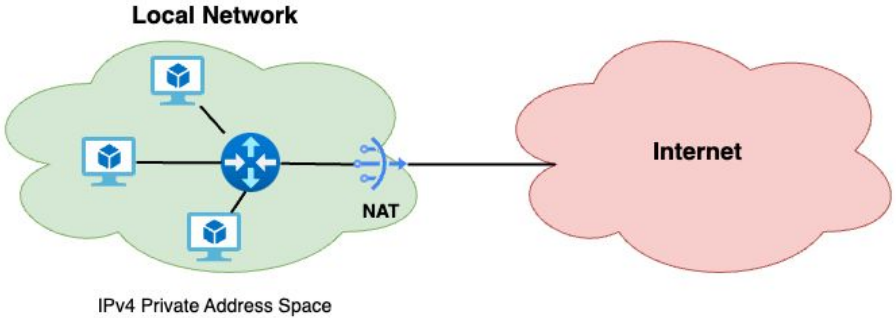
- IPv6 addresses are 128-bit long
- Similarly to IPv4:
  - Addresses can be aggregated into prefixes (e.g. for routing)
  - Multiple address types (unicast, multicast…)
  - Multiple address scopes (link-local, unique-local, global)
- But IPv6 hosts typically employ **multiple addresses** of different properties
- IPv6 subnets are typically a /64

IPv6 Unique Local Addresses (ULAs)

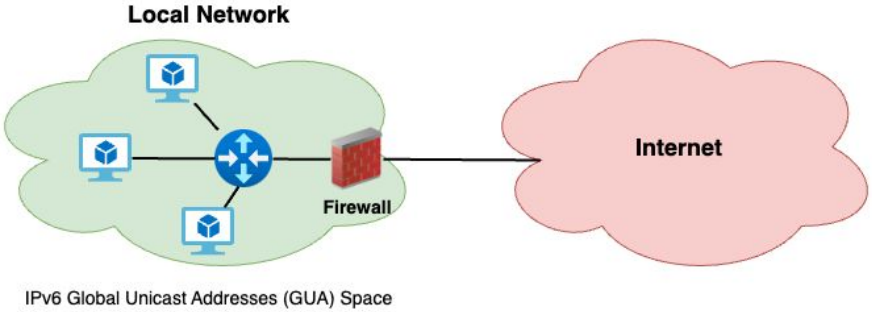
- Provide private address space for IPv6
  - IPv6-version of RFC1918 addresses
  - ULA prefix is randomized to avoid address space overlaps
  - Controversial feature in the IPv6 addressing architecture
- Expected use case:
  - Internal communications within a network
  - **Not** meant to be used in combination with IPv6 NAT

Deployment models: IPv4 vs. IPv6

IPv4



IPv6



Kubernetes & IPv6

+

Kubernetes API: IPv4


```
apiVersion: v1
kind: Node
metadata:
  name: node1
spec:
  podCIDR: 10.0.0.0/24
...

status:
  addresses:
  - address: 192.168.0.5
    type: InternalIP
```

```
apiVersion: v1
kind: Pod
metadata:
  name: app-http
  namespace: default
spec:
  containers:
  - name: app
    image: myapp:0.1
…
status:
  phase: Running
  podIP: 10.0.0.5
```

```
apiVersion: v1
kind: Service
metadata:
  name: lb-app
  namespace: default
spec:
  clusterIP: 10.96.0.13
...
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 196.23.45.23
```

Kubernetes API: IPv6

KubeCon CloudNativeCon Europe 2023

```
apiVersion: v1
kind: Node
metadata:
  name: node1
spec:
  podCIDR: fd00:1234:1::/64
...

status:
  addresses:
  - address: fd01:789a:1::1
    type: InternalIP
```

```
apiVersion: v1
kind: Pod
metadata:
  name: app-http
  namespace: default
spec:
  containers:
  - name: app
    image: myapp:0.1
...
status:
  phase: Running
  podIP: fd00:1234:1::1
```

```
apiVersion: v1
kind: Service
metadata:
  name: lb-app
  namespace: default
spec:
  clusterIP: fd12:789a:1::1
...
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 2001:db8:7:8::4
```

Kubernetes API: Dual Stack

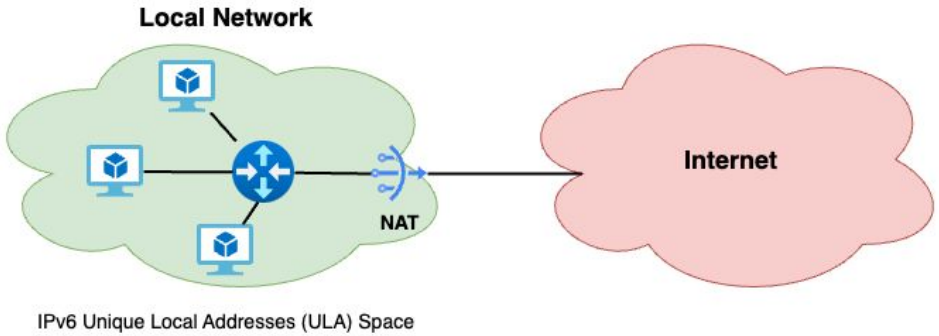

```
apiVersion: v1
kind: Node
metadata:
  name: node1
spec:
  podCIDR: 10.0.0.0/24
  podCIDRs:
    -    10.0.0.0/24
    -    fd00:1234:1::/64
...
 status:
    addresses:
    - address: fd01:789a:1::1
      type: InternalIP
    - address: 192.168.0.1
      type: InternalIP
```

```
apiVersion: v1
kind: Pod
metadata:
  name: app-http
  namespace: default
spec:
  containers:
  - name: app
    image: myapp:0.1
…
status:
  phase: Running
  podIP: 10.0.0.5
  podIPs:
    -    10.0.0.5
    -    fd00:1234:1::1
```

```
apiVersion: v1
kind: Service
metadata:
  name: lb-app
  namespace: default
spec:
  clusterIP: 10.96.0.13
  clusterIPs:
    -    10.96.0.13
    -    fd12:789a:1::1
...
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 192.168.21.8
    - ip: 2001:db8:7:8::4
```

IPv6 support in Kubernetes

- One common implementation strategy is replicate the IPv4 architecture with IPv6:
  - Use Unique Local Addresses (private address space) virtually everywhere
  - Expose services with a global (public) address via a load-balancer
- Well-understood model: IPv6 architecture matches the IPv4 counterpart



Kubernetes & IPv6: The Future Ahead

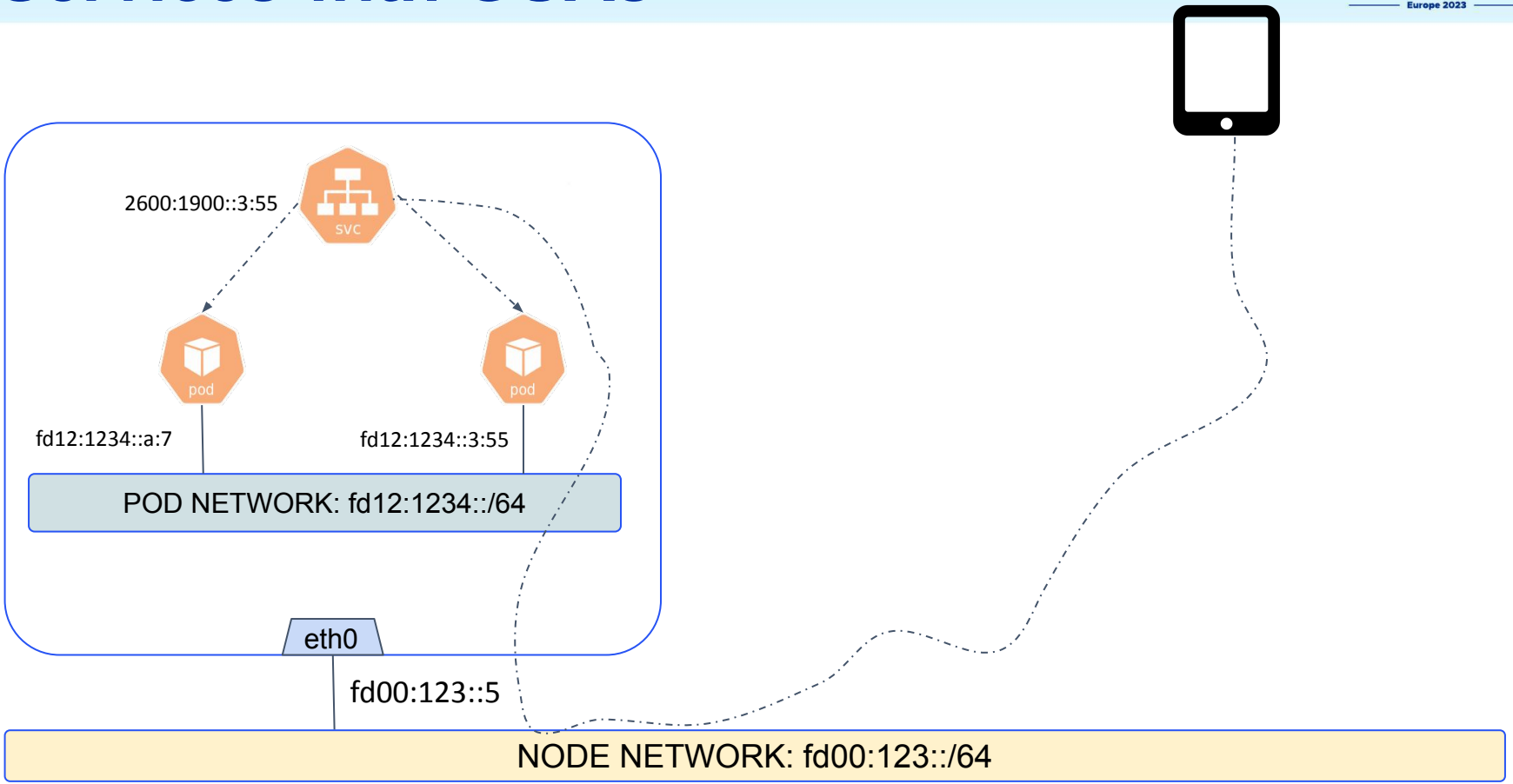
IPv6 support in Kubernetes (revisited)

- Kubernetes IPv6 support mimics its IPv4 counterpart
- Are we missing a chance to leverage IPv6 capabilities?
- How could we possibly evolve IPv6 support?
- What if we were to leverage Global Unicast Addresses?

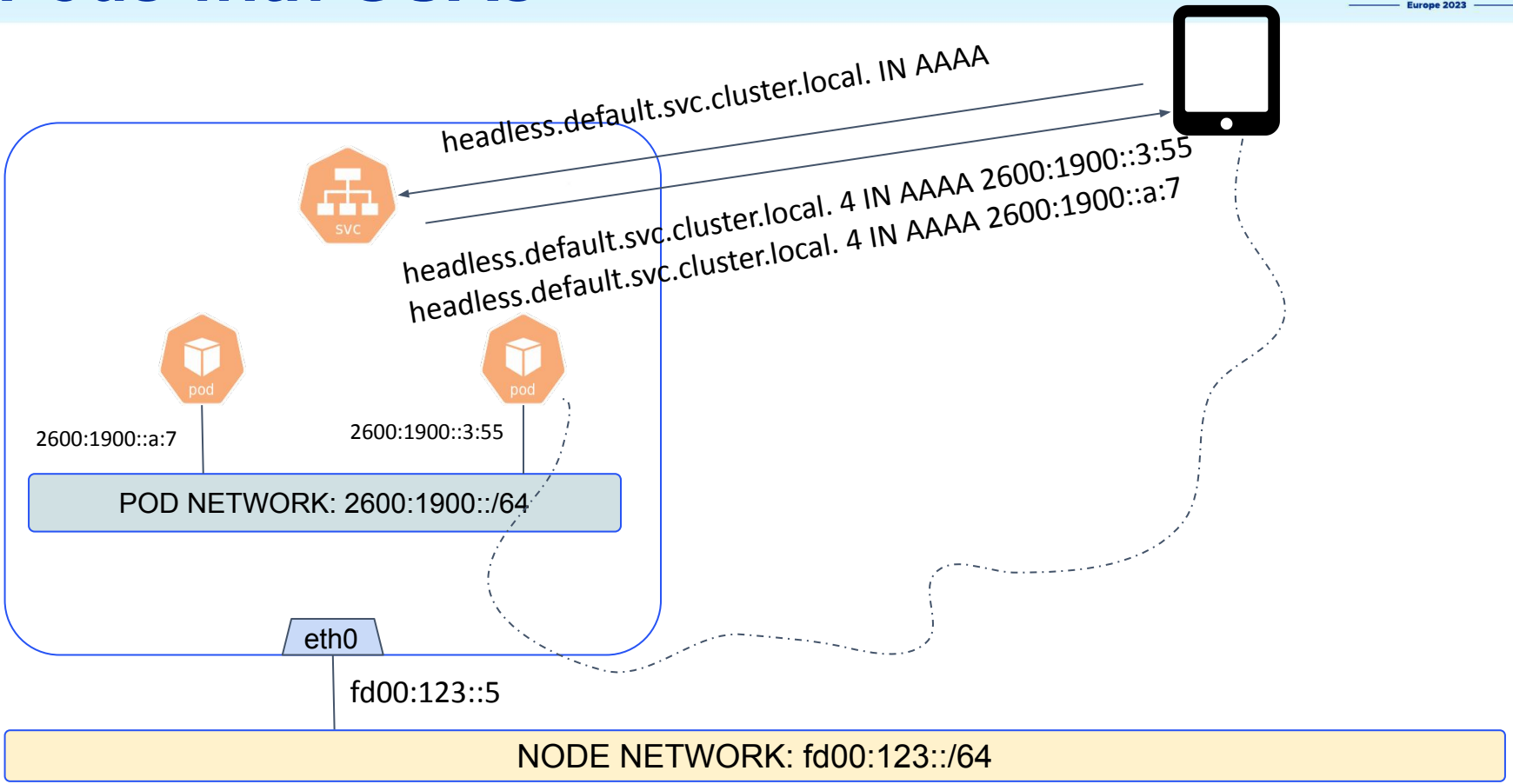
Leveraging global IPv6 addresses

- Services with Global Unicast Addresses (GUAs)
  - Remove one layer of translation!
- Pods with GUAs, and leverage the DNS
  - Remove two layers of translation!

Services with GUAs



Pods with GUAs




TO BE
CONTINUED...


TO BE
CONTINUED...


To be continued...


KubeCon
CloudNativeCon
Europe 2023
Questions?

Please scan the QR Code above
to leave feedback on this session