# Navigating the Networking Maze of Kubernetes

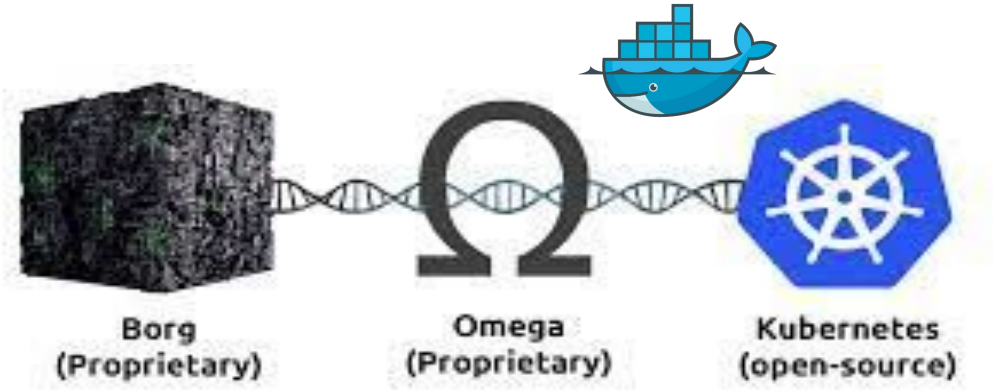
A Journey of Discovery, Confusion, and (Hopefully) Enlightenment

aojea@google.com

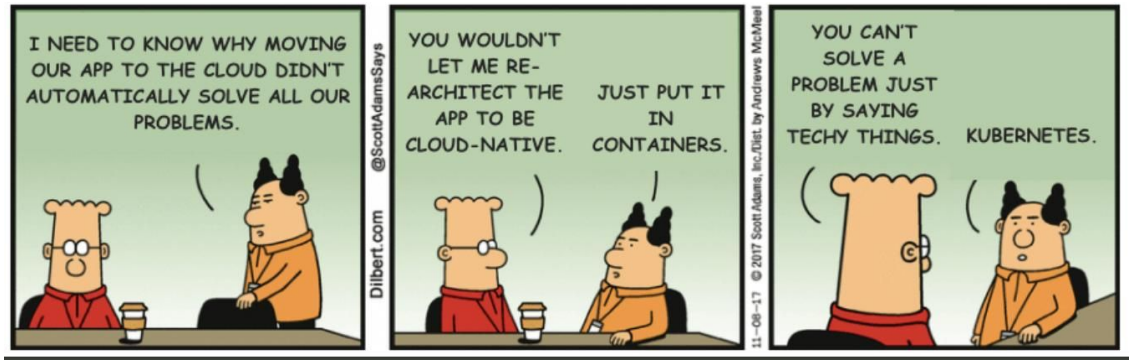
FOSDEM 2024

Google

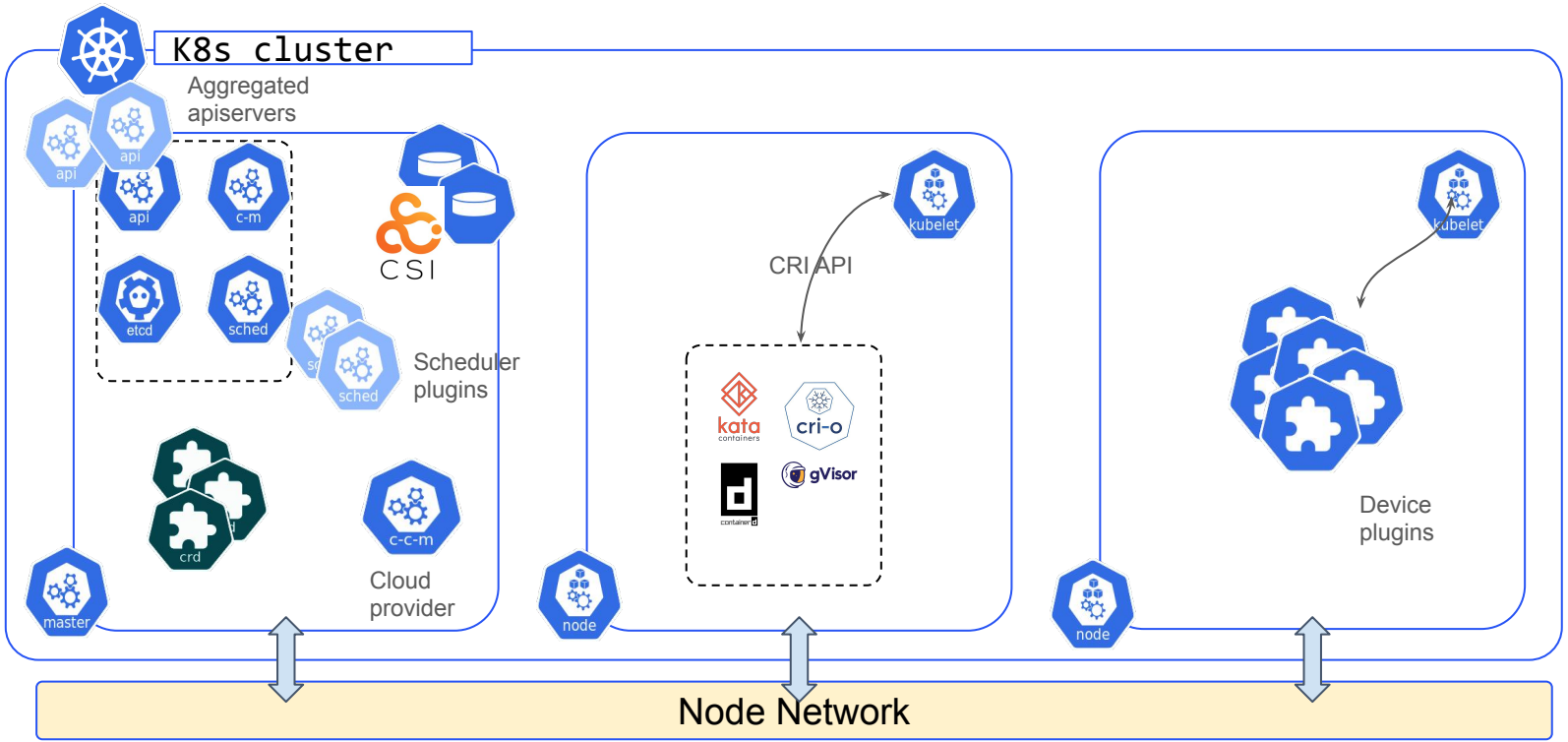
How it started




How it is going



# Kubernetes Architecture: extensible and pluggable



# Kubernetes Architecture: API + Semantics

## 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
```

community / contributors / devel / sig-testing / **e2e-tests.md**

lunarwhite Fix broken links in contributors guide

Preview | Code | Blame | 786 lines (604 loc) · 35 KB

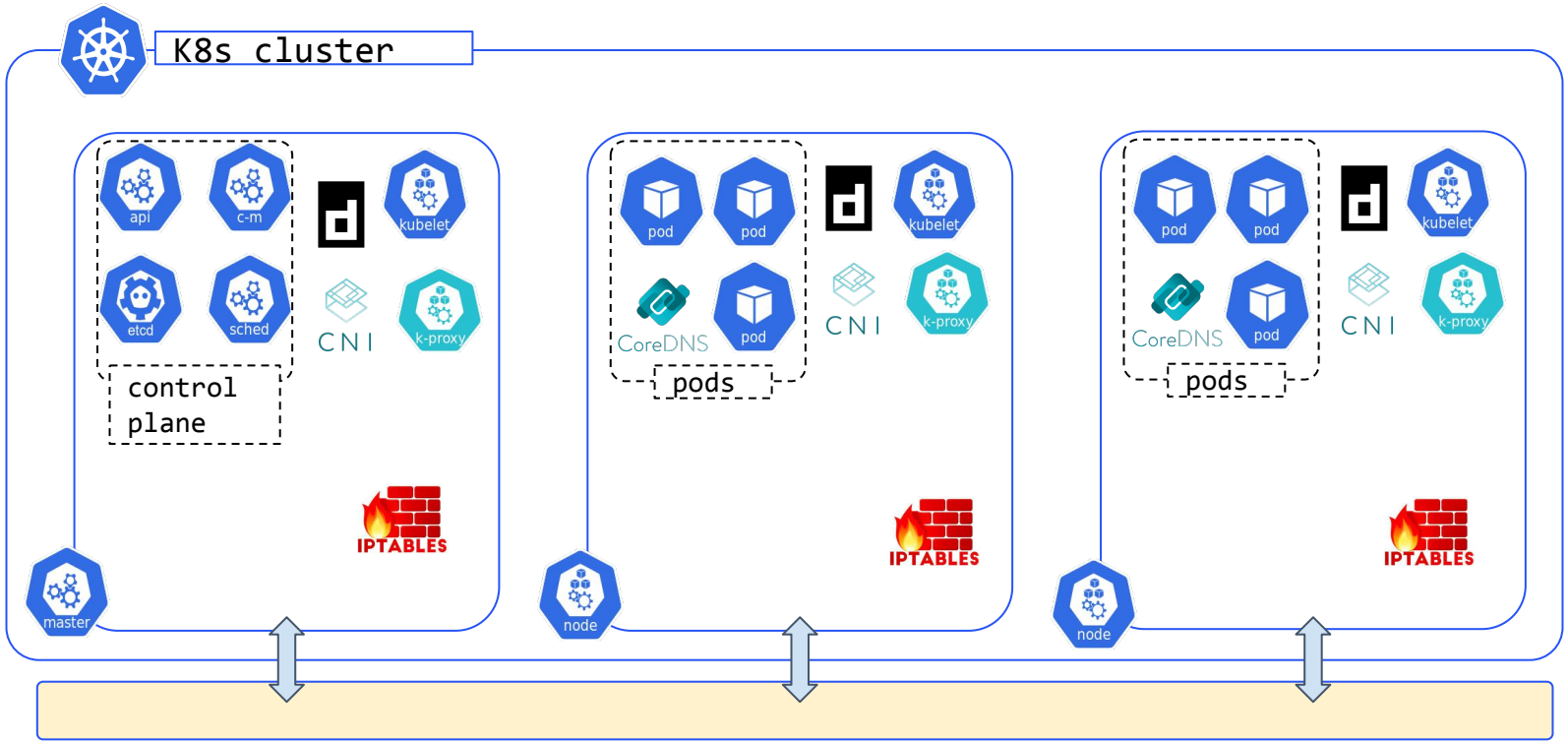
### End-to-End Testing in Kubernetes

Table of Contents

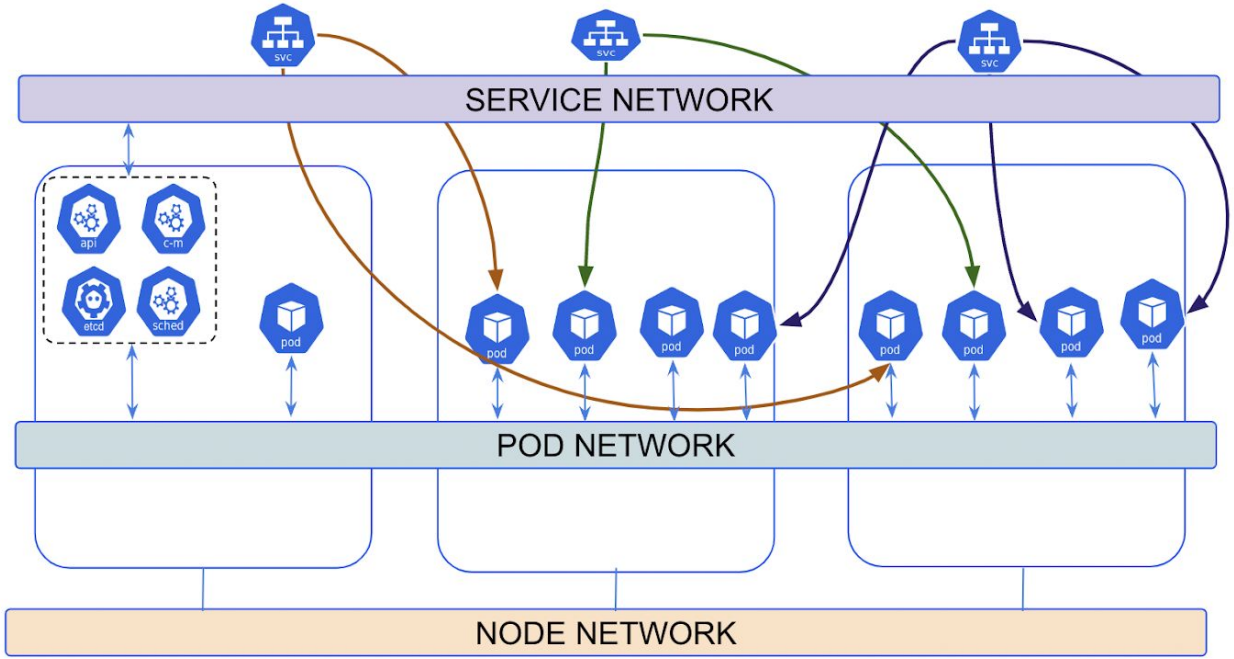


certified kubernetes

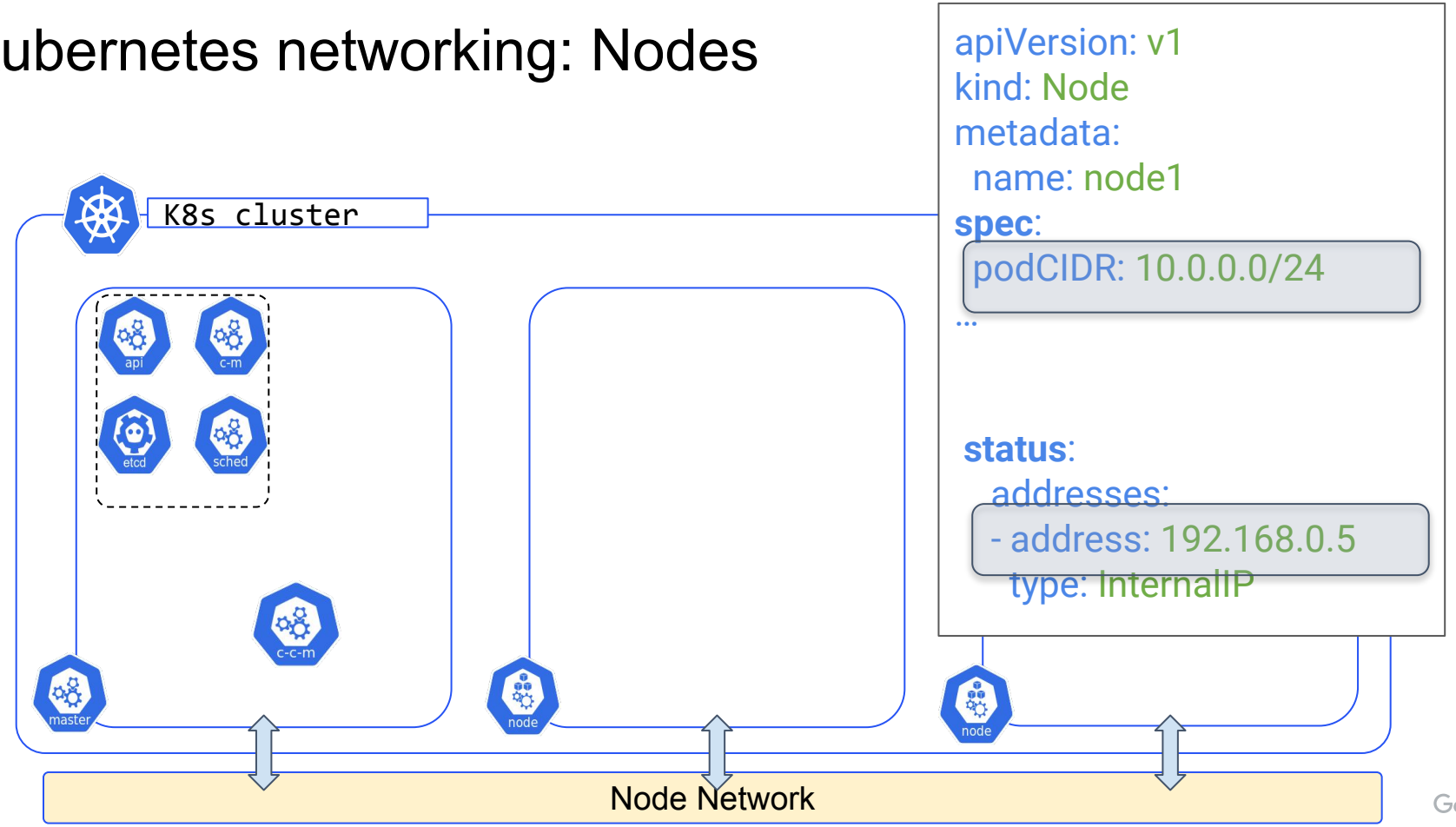
# Kubernetes Implementation: one of multiple



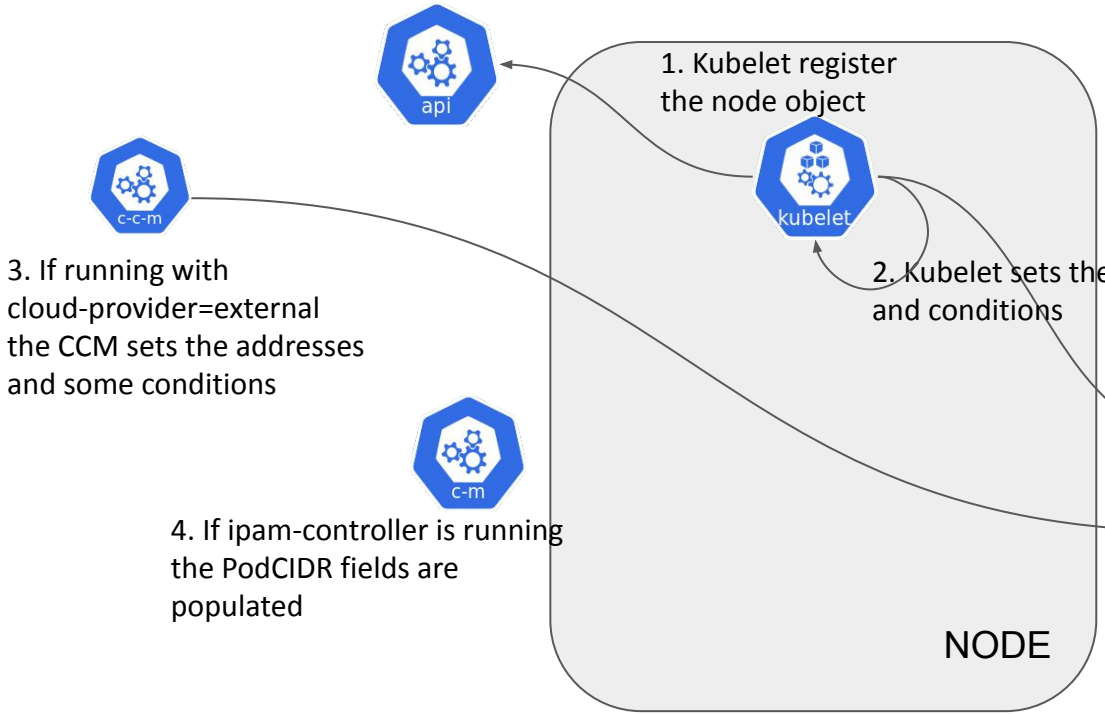
# Kubernetes Networking: end to end principle



# Kubernetes networking: Nodes



# Kubernetes networking: Nodes

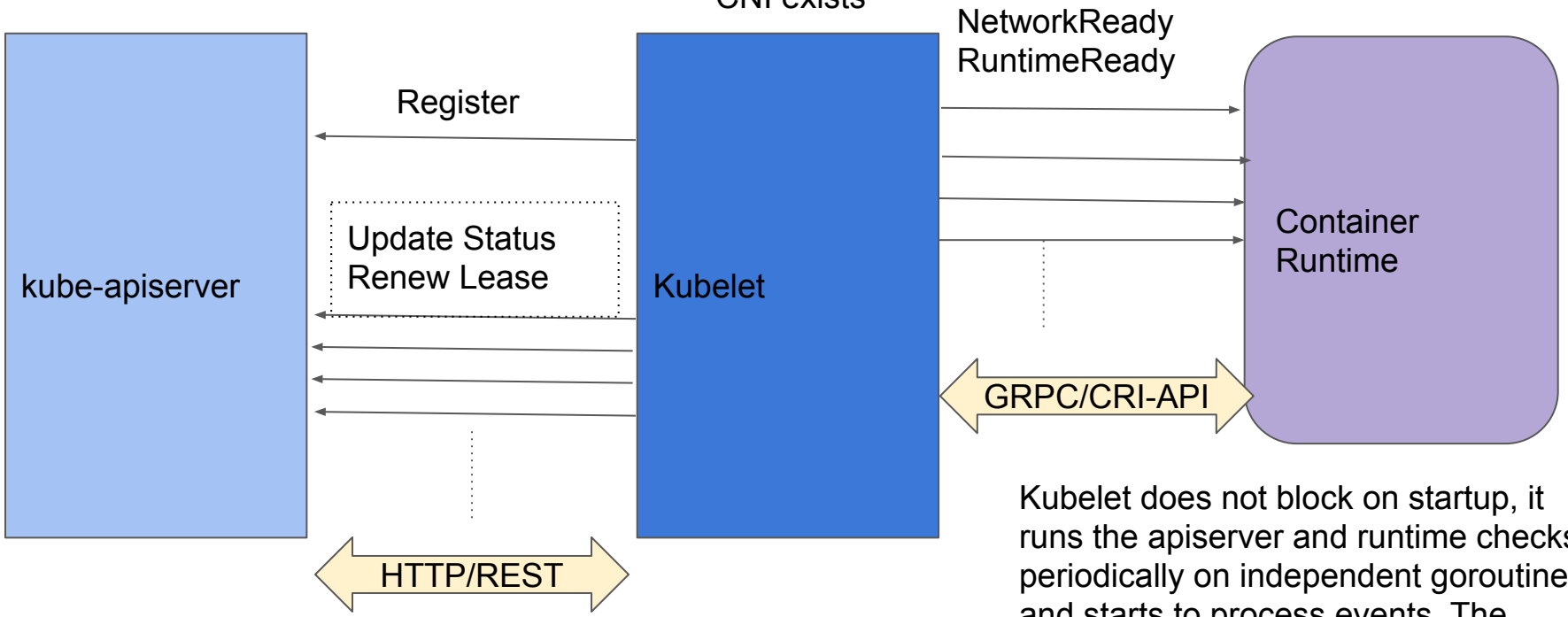



```
apiVersion: v1
kind: Node
metadata:
  name: node1
spec:
  podCIDR: 10.0.0.0/24
...

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

# Node Initialization

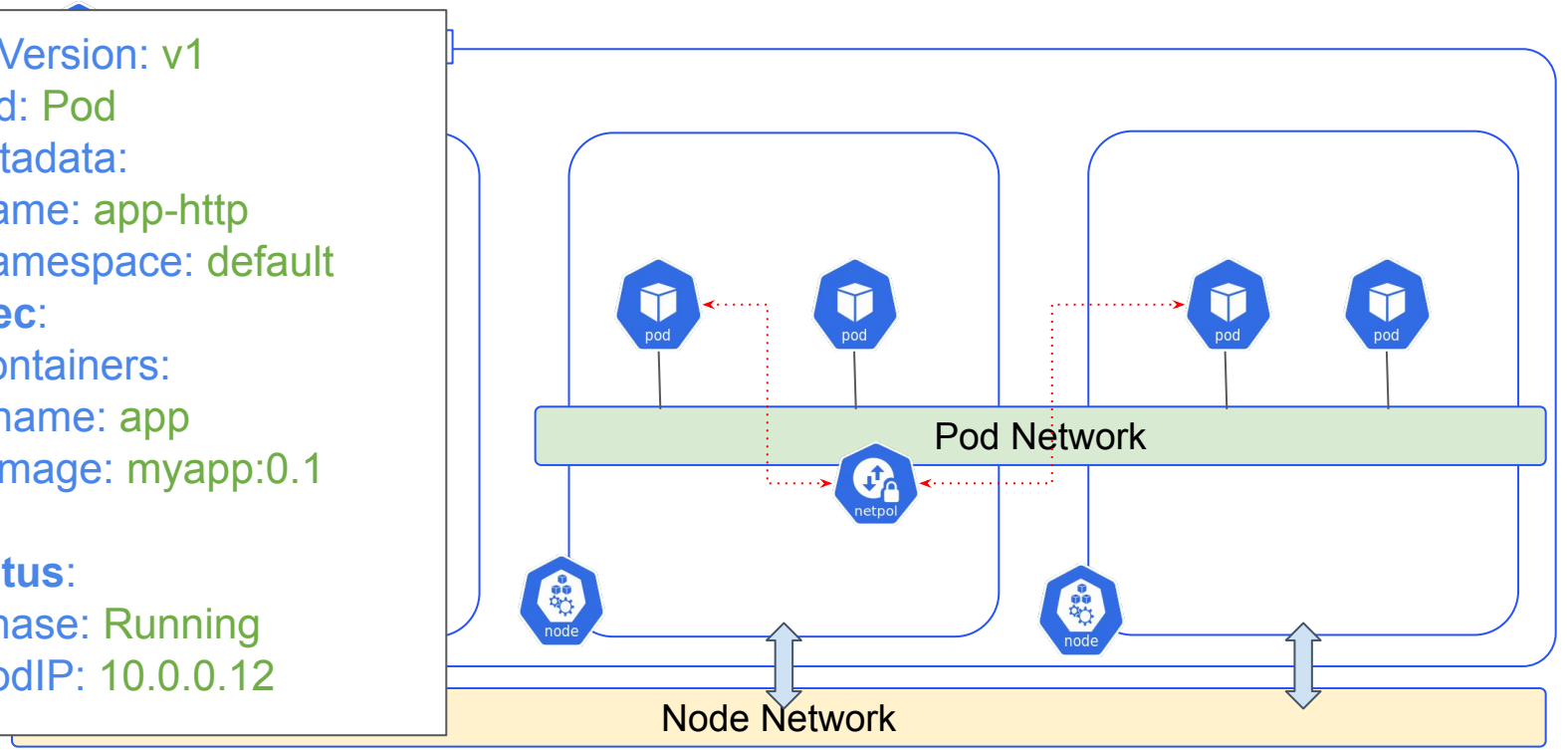
One of the runtime checks is “NetworkReady”, this is implemented on the container runtimes just as a check that a configuration file for the CNI exists



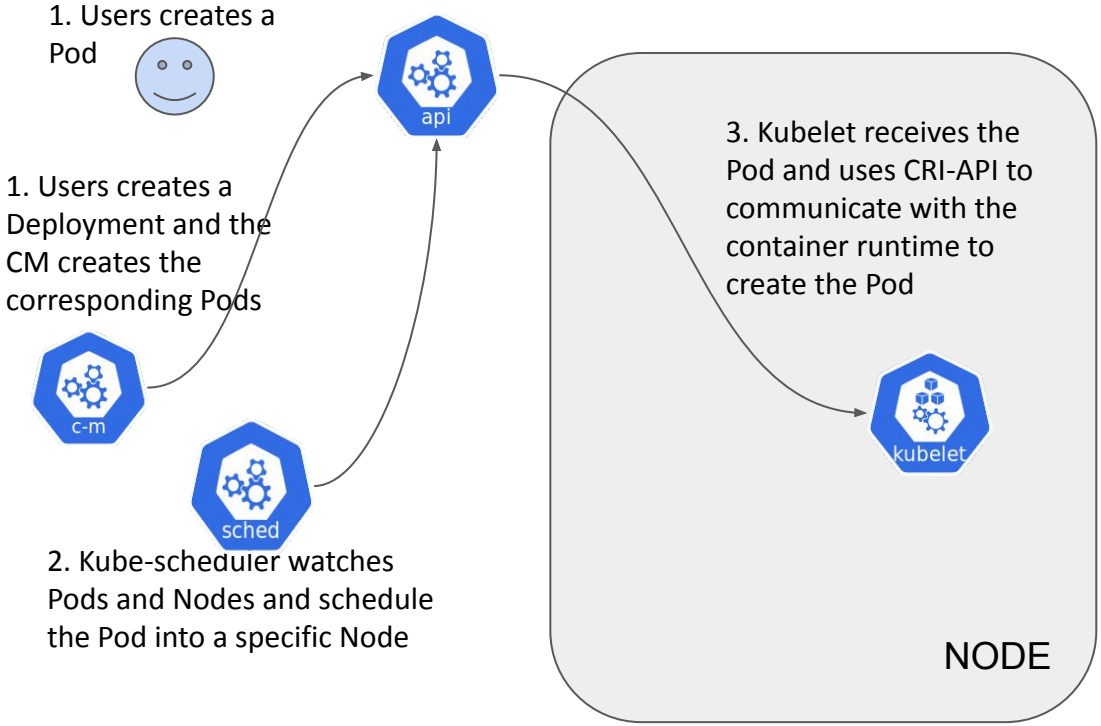

Kubelet does not block on startup, it runs the apiserver and runtime checks periodically on independent goroutines and starts to process events. The Node is not declared Ready until all the checks are OK

# Kubernetes networking: Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: app-http
  namespace: default
spec:
  containers:
  - name: app
    image: myapp:0.1
…
status:
  phase: Running
  podIP: 10.0.0.12
```

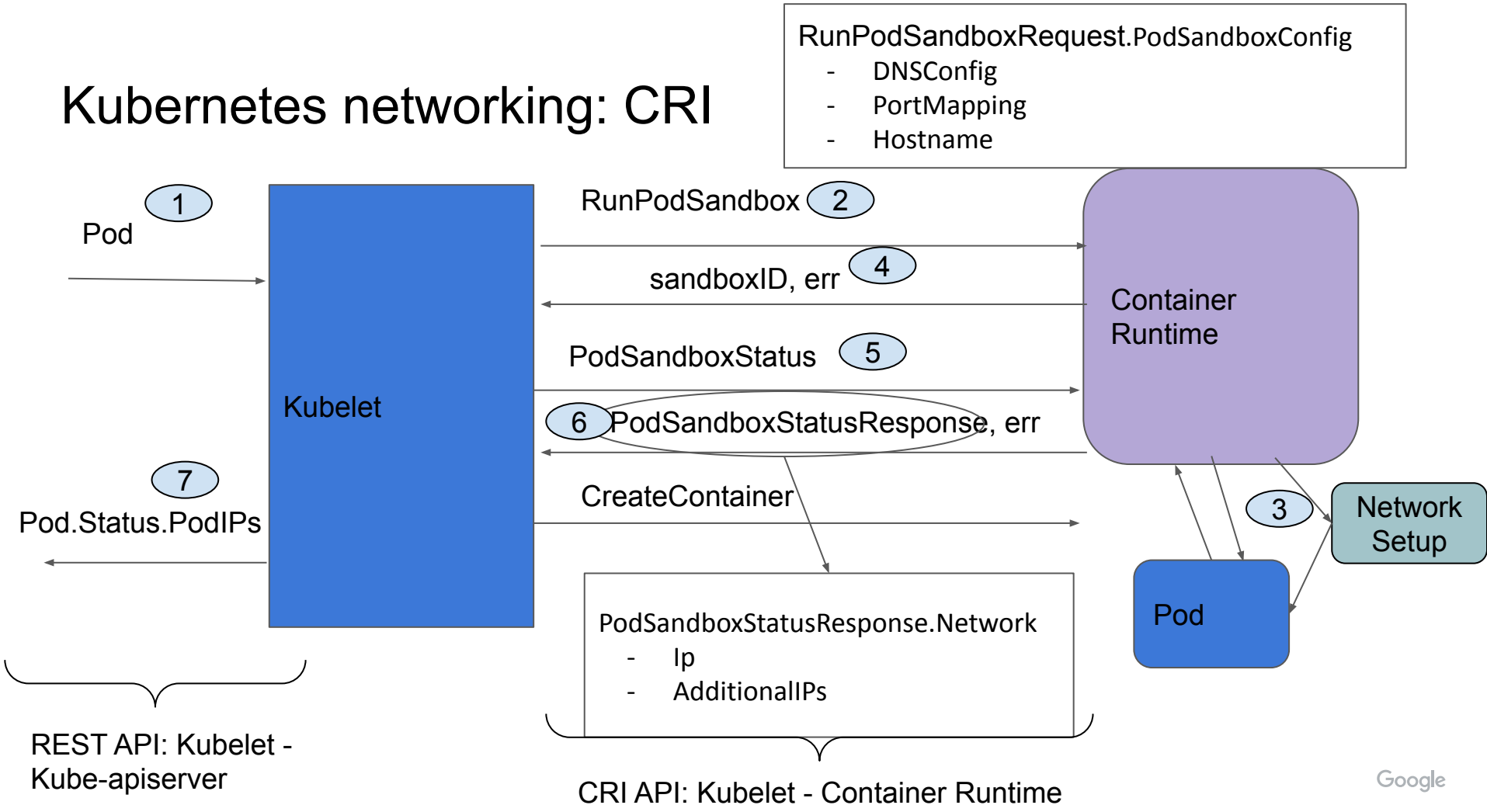


# Kubernetes networking: Pods

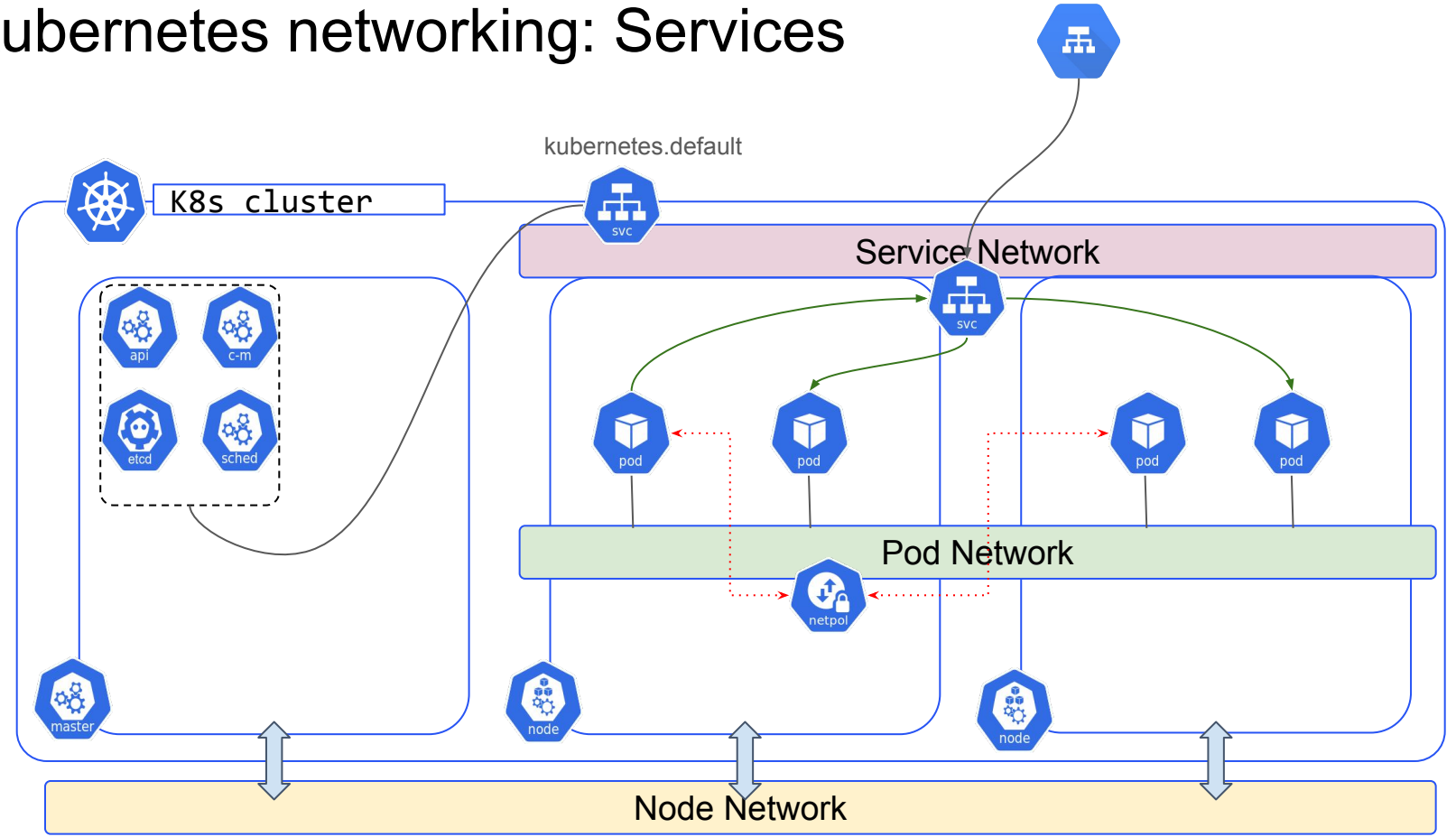



```
apiVersion: v1
kind: Pod
metadata:
  name: app-http
  namespace: default
spec:
  containers:
  - name: app
    image: myapp:0.1
…
status:
  phase: Running
  podIP: 10.0.0.12
```

# Kubernetes networking: CRI



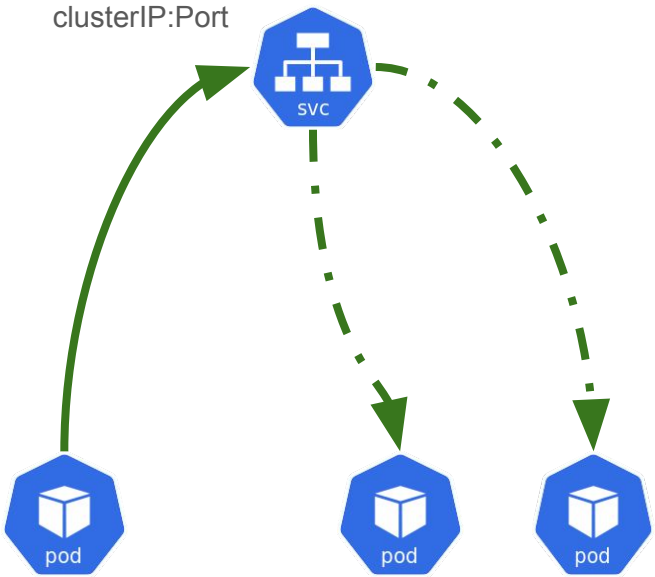
# Kubernetes networking: Services



# Kubernetes Services: ClusterIP

```yaml
apiVersion: v1
kind: Service
metadata:
  name: service
spec:
  clusterIP: 10.96.0.1
  clusterIPs:
  - 10.96.0.1
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: 6443
  type: ClusterIP
```

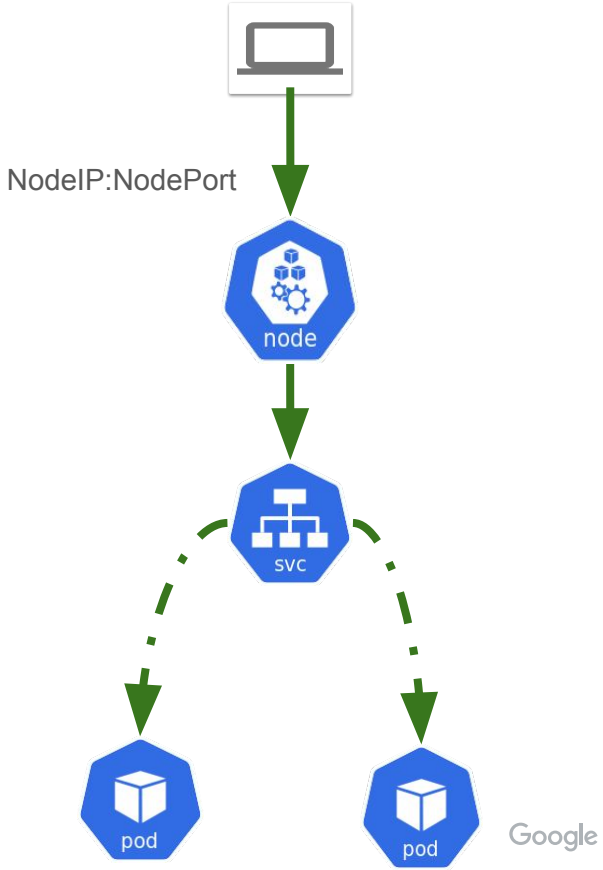
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deployment
  labels:
    app: MyApp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: MyApp
  template:
    metadata:
      labels:
        app: MyApp
    spec:
      terminationGracePeriodSeconds: 30
      containers:
      - name: agnhost
        image:
k8s.gcr.io/e2e-test-images/agnhost:2.39
        args:
          - netexec
          - --http-port=80
```



# Kubernetes Services: NodePort

```yaml
apiVersion: v1
kind: Service
metadata:
  name: service
spec:
  clusterIP: 10.96.0.1
  clusterIPs:
  - 10.96.0.1
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: 6443
    nodePort: 31023
  type: NodePort
```

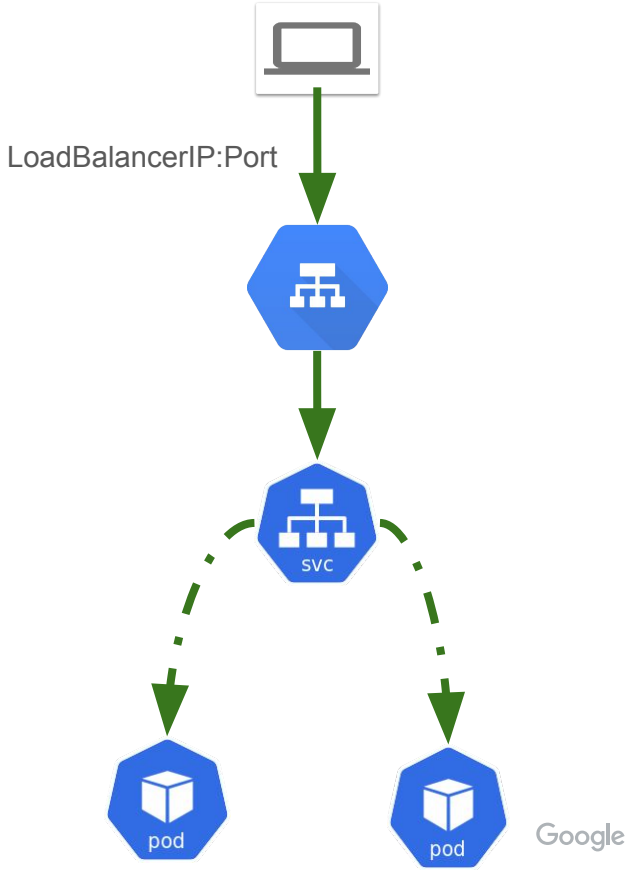
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deployment
  labels:
    app: MyApp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: MyApp
  template:
    metadata:
      labels:
        app: MyApp
    spec:
      terminationGracePeriodSeconds: 30
      containers:
      - name: agnhost
        image:
k8s.gcr.io/e2e-test-images/agnhost:2.39
        args:
          - netexec
          - --http-port=80
```



# Kubernetes Services: Load Balancer

```yaml
apiVersion: v1
kind: Service
metadata:
  name: service
spec:
  clusterIP: 10.96.0.1
  clusterIPs:
  - 10.96.0.1
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: 6443
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    -   ip: 202.34.23.12
```

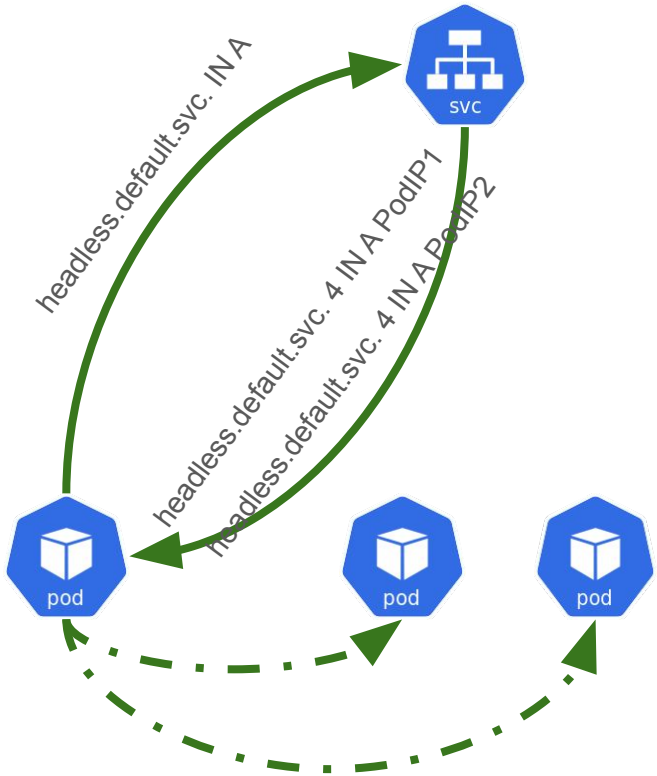
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deployment
  labels:
    app: MyApp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: MyApp
  template:
    metadata:
      labels:
        app: MyApp
    spec:
      terminationGracePeriodSeconds: 30
      containers:
      - name: agnhost
        image:
k8s.gcr.io/e2e-test-images/agnhost:2.39
        args:
          - netexec
          - --http-port=80
```



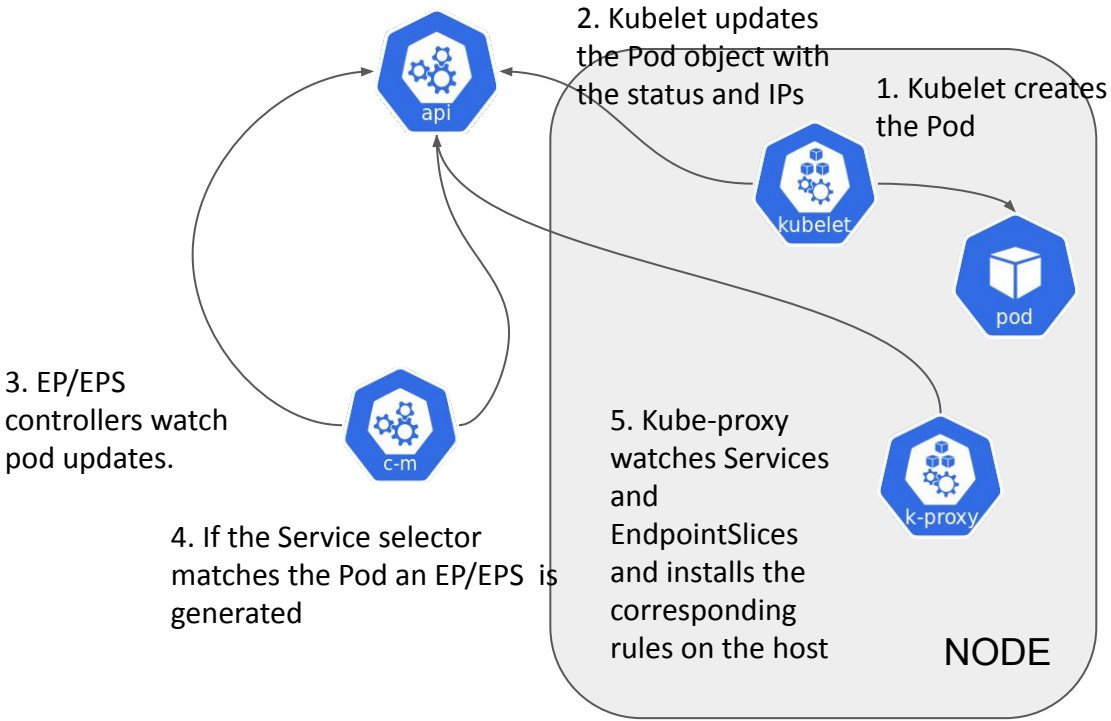
# Kubernetes Services: Headless

```yaml
apiVersion: v1
kind: Service
metadata:
  name: headless
spec:
  clusterIP: None
  clusterIPs:
  - None
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: https
    port: 443
    protocol: TCP
    targetPort: 6443
  type: ClusterIP
```

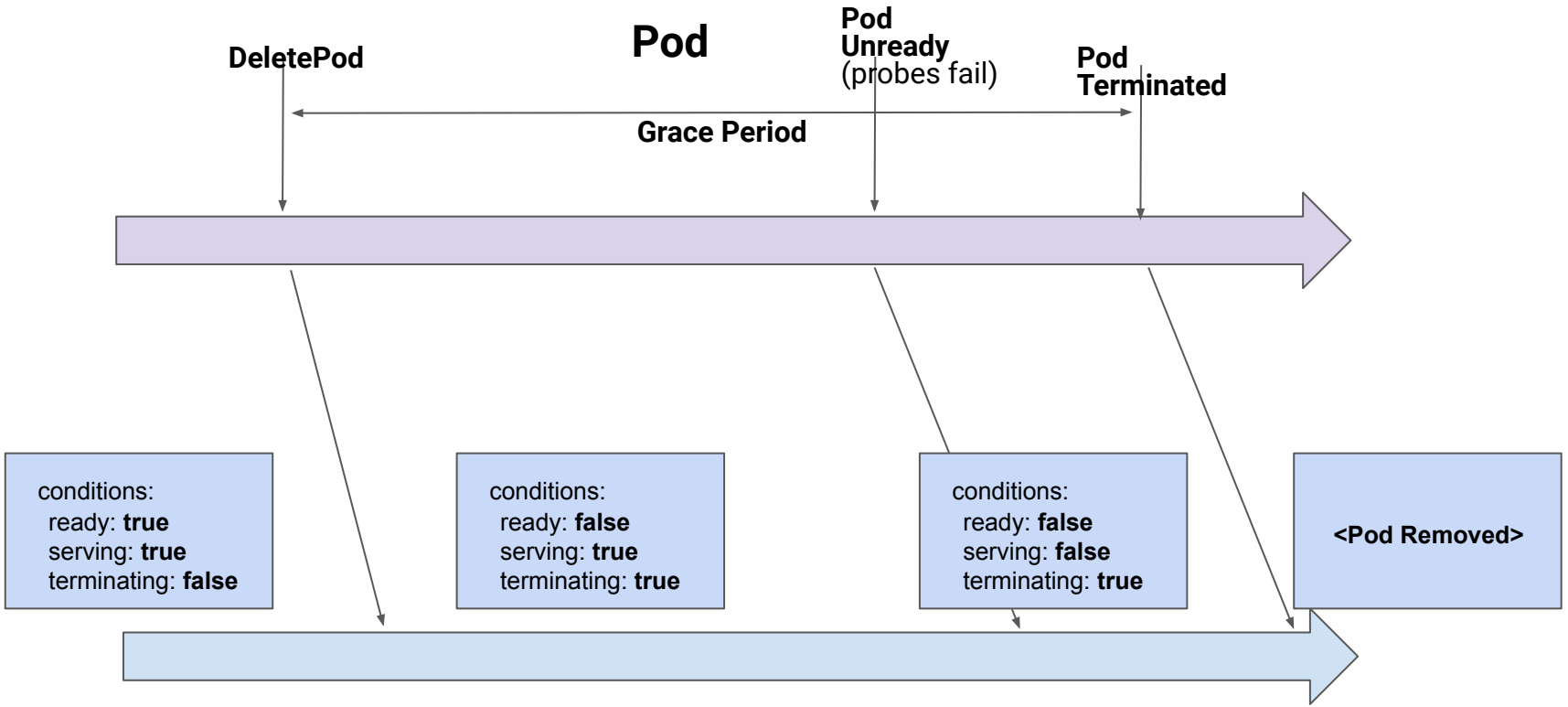
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deployment
  labels:
    app: MyApp
spec:
  replicas: 2
  selector:
    matchLabels:
      app: MyApp
  template:
    metadata:
      labels:
        app: MyApp
    spec:
      terminationGracePeriodSeconds: 30
      containers:
      - name: agnhost
        image:
k8s.gcr.io/e2e-test-images/agnhost:2.39
        args:
          - netexec
          - --http-port=80
```

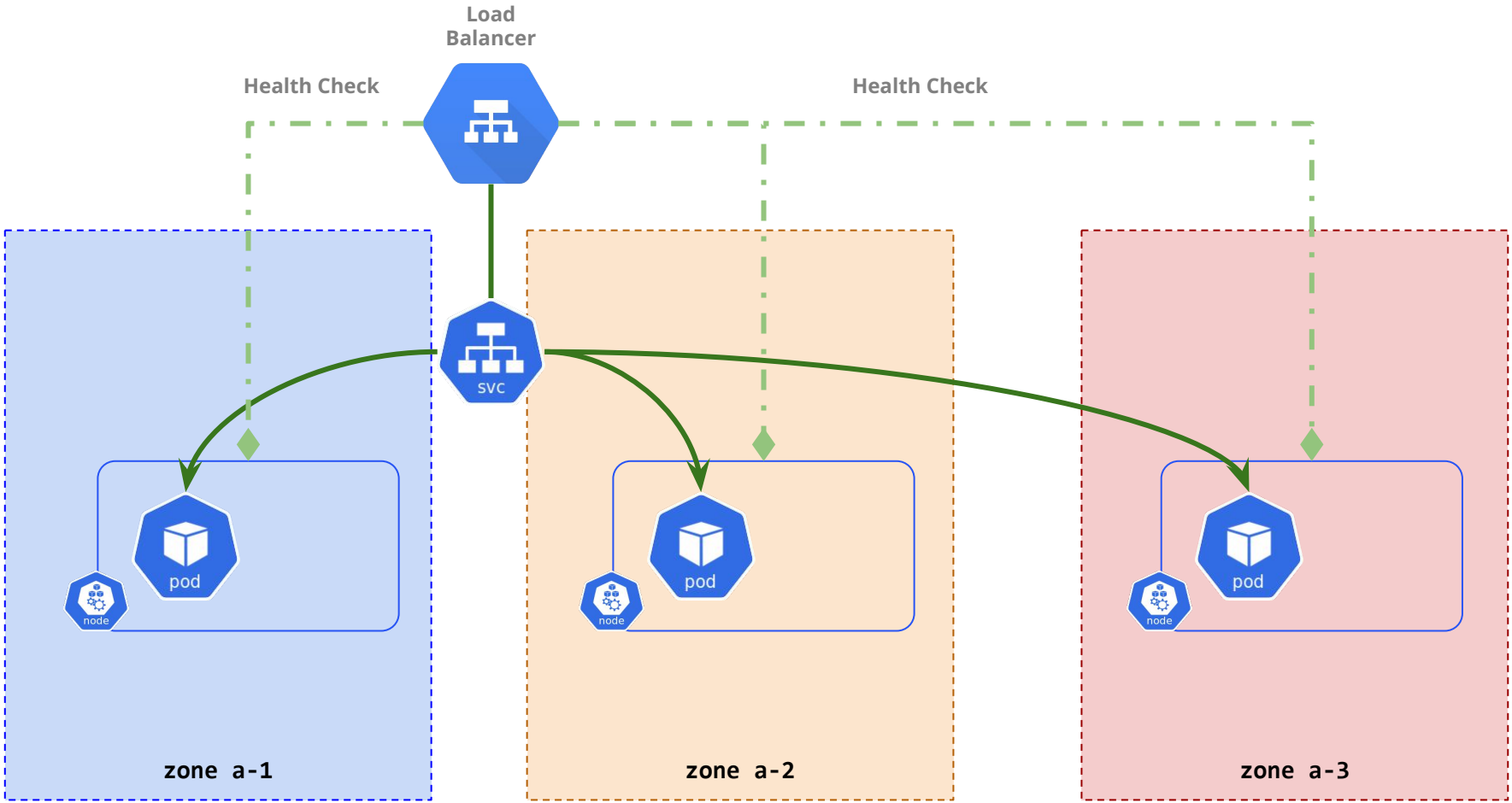


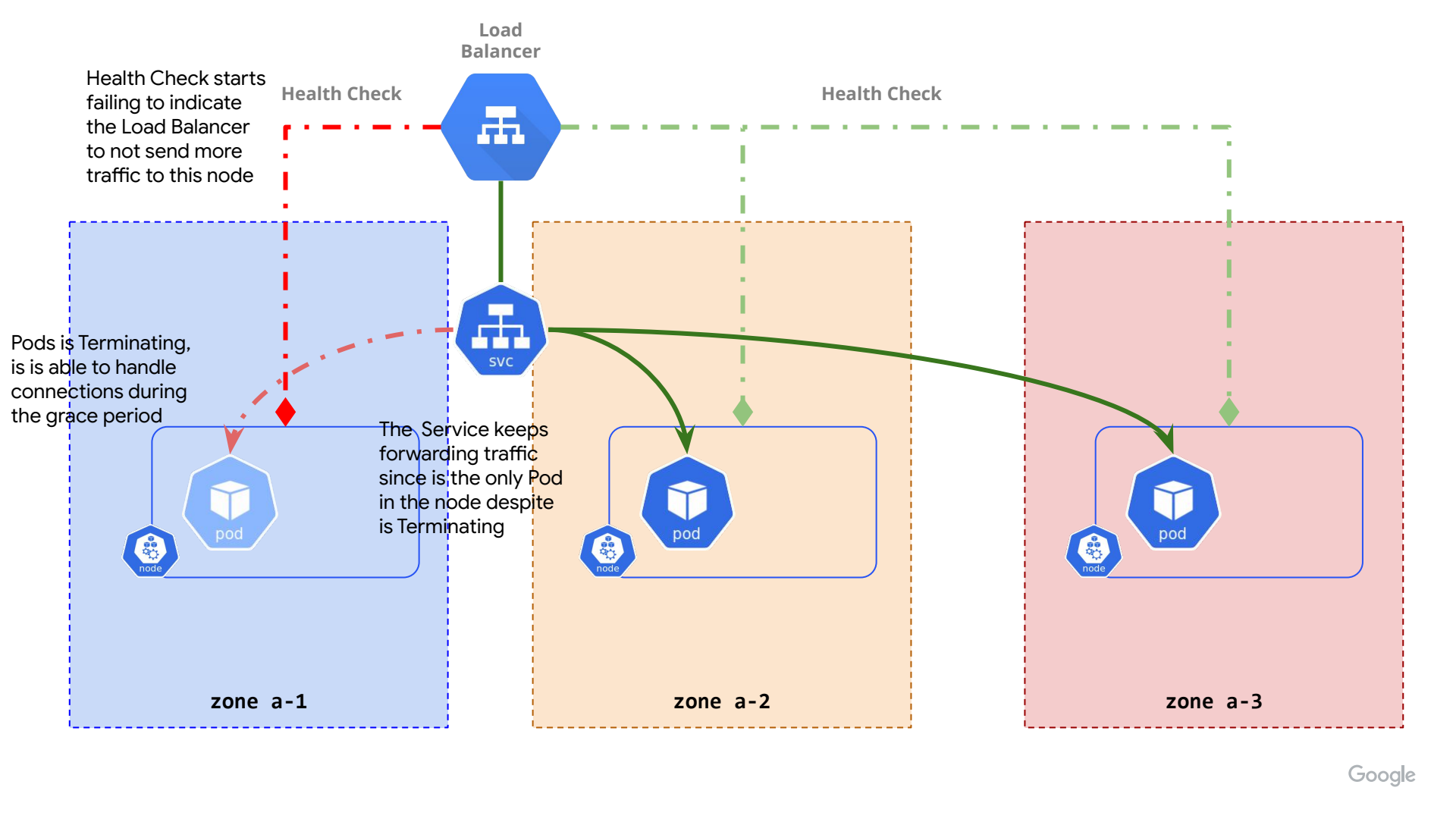
# Kubernetes Services

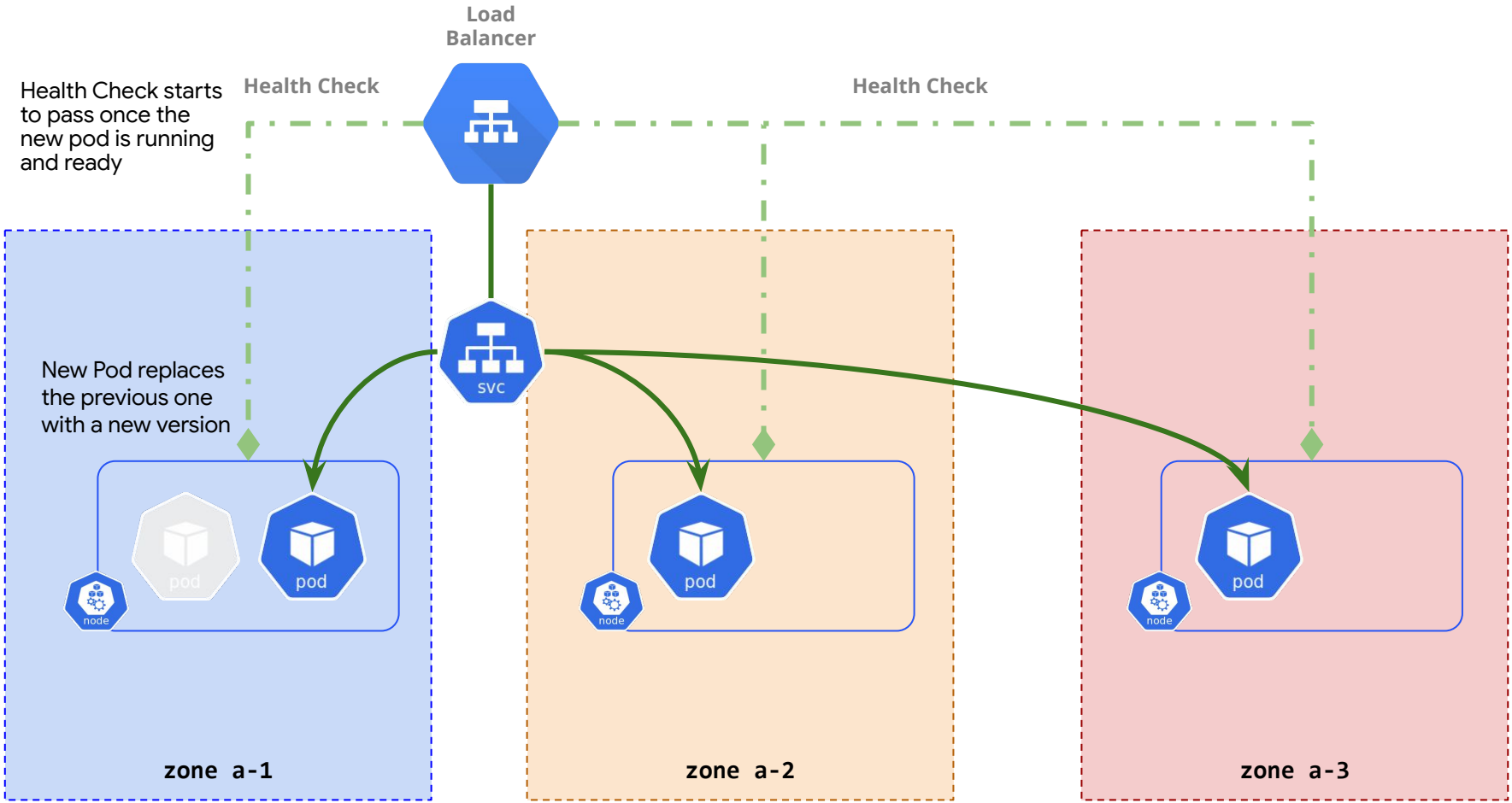



```
apiVersion: v1
kind: Service
metadata:
  name: lb-app
  namespace: default
spec:
  clusterIP: 10.96.0.13
  selector:
    app: MyApp
  type: ClusterIP
  ports:
  - name: port8080
    port: 8080
    protocol: TCP
    targetPort: 8080
```


Pod
DeletePod
Pod
Unready
(probes fail)
Pod
Terminated
Grace Period
conditions:
ready: true
serving: true
terminating: false
conditions:
ready: false
serving: true
terminating: true
conditions:
ready: false
serving: false
terminating: true
<Pod Removed>
EndpointSlice

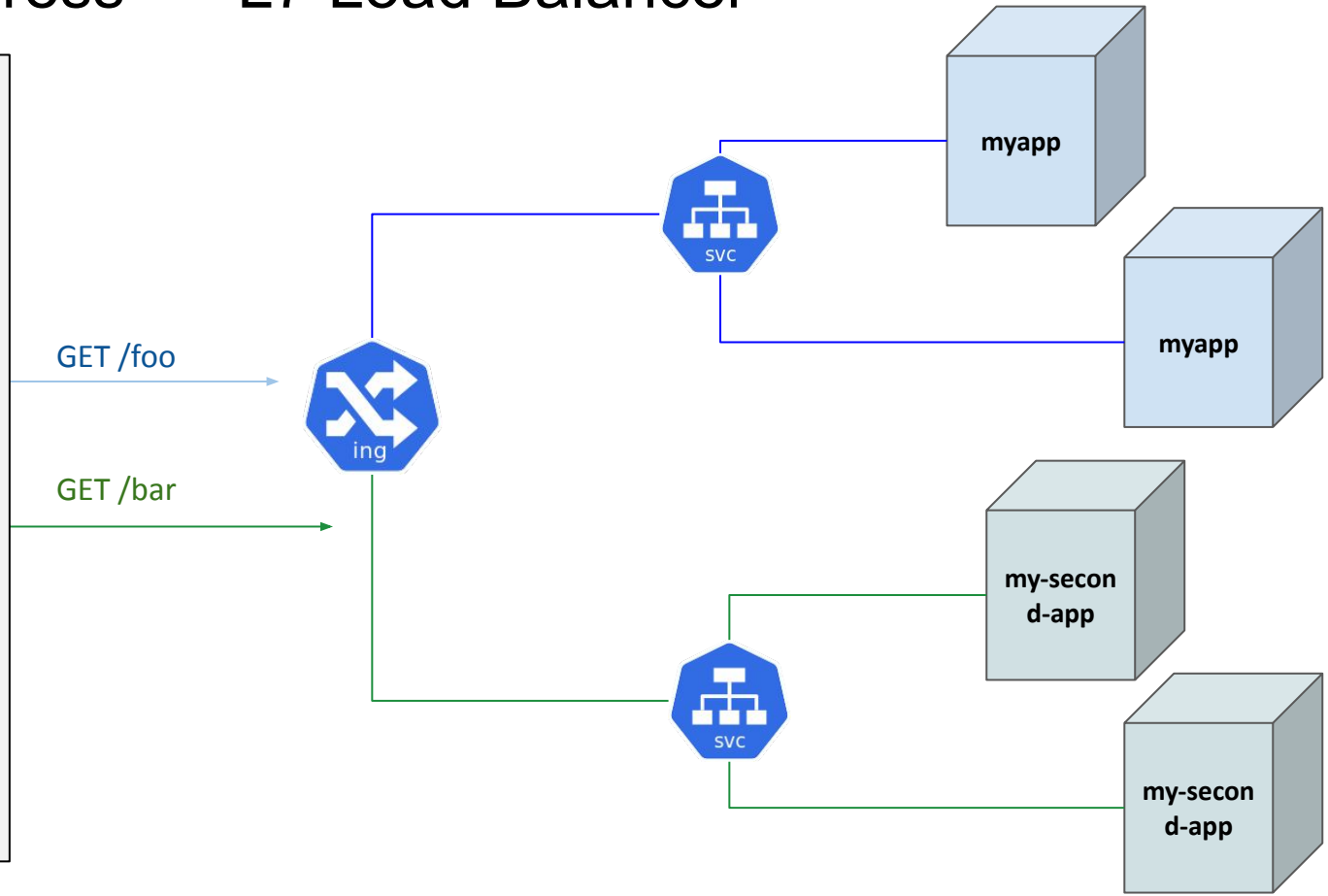

Load
Balancer
Health Check
Health Check
SVC
pod
node
pod
node
pod
node
zone a-1
zone a-2
zone a-3


Load Balancer
Health Check starts failing to indicate the Load Balancer to not send more traffic to this node
Health Check
Health Check
SVC
Pods is Terminating, is is able to handle connections during the grace period
The Service keeps forwarding traffic since is the only Pod in the node despite is Terminating
pod
node
pod
node
pod
node
zone a-1
zone a-2
zone a-3

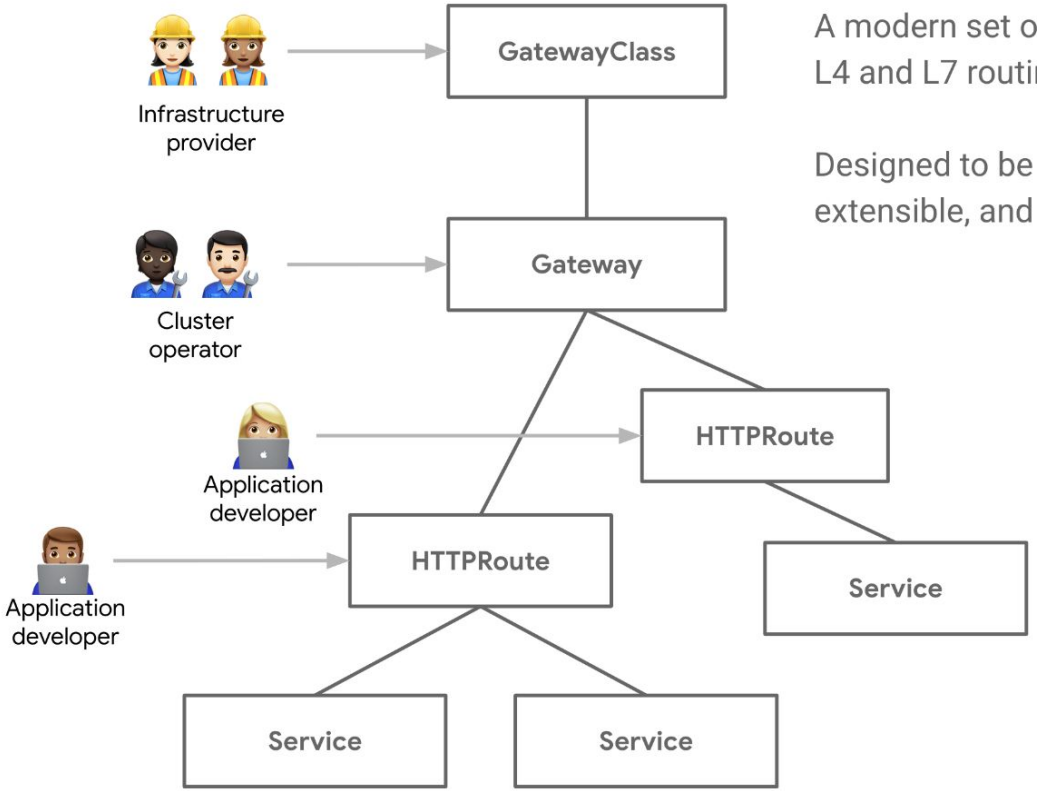

Load Balancer
Health Check starts to pass once the new pod is running and ready
Health Check
Health Check
New Pod replaces the previous one with a new version
svc
pod
pod
node
pod
node
pod
node
zone a-1
zone a-2
zone a-3

# Kubernetes Ingress == L7 Load Balancer

```yaml
apiVersion: v1
kind: Ingress
metadata:
  name: minimal-ingress
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /foo
        pathType: Prefix
        backend:
          service:
            name: myapp
            port:
              number: 80
      - path: /bar
        pathType: Prefix
        backend:
          service:
            name: my-second-app
            port:
              number: 80
```

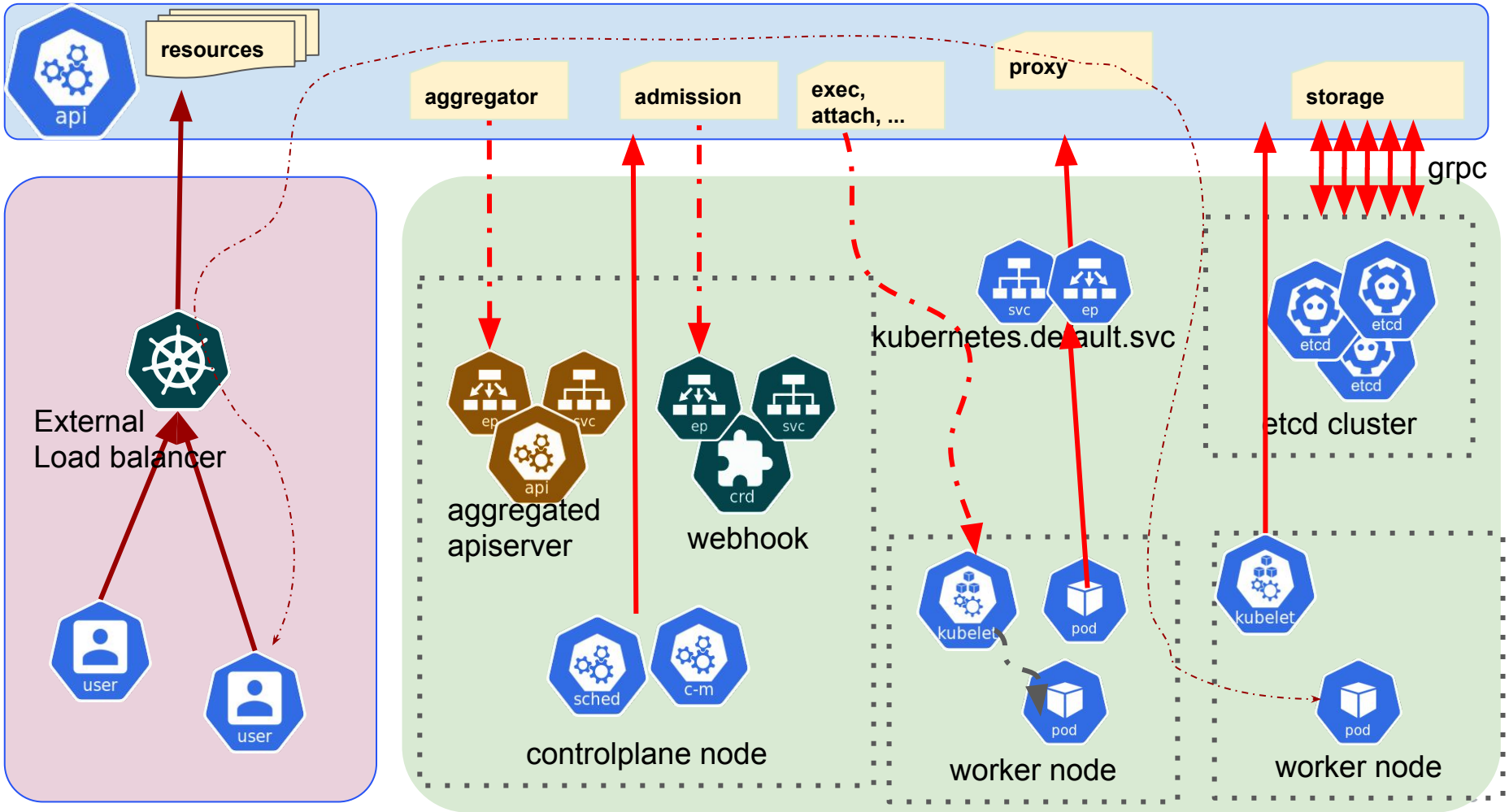


# Gateway API aims to standardize the space

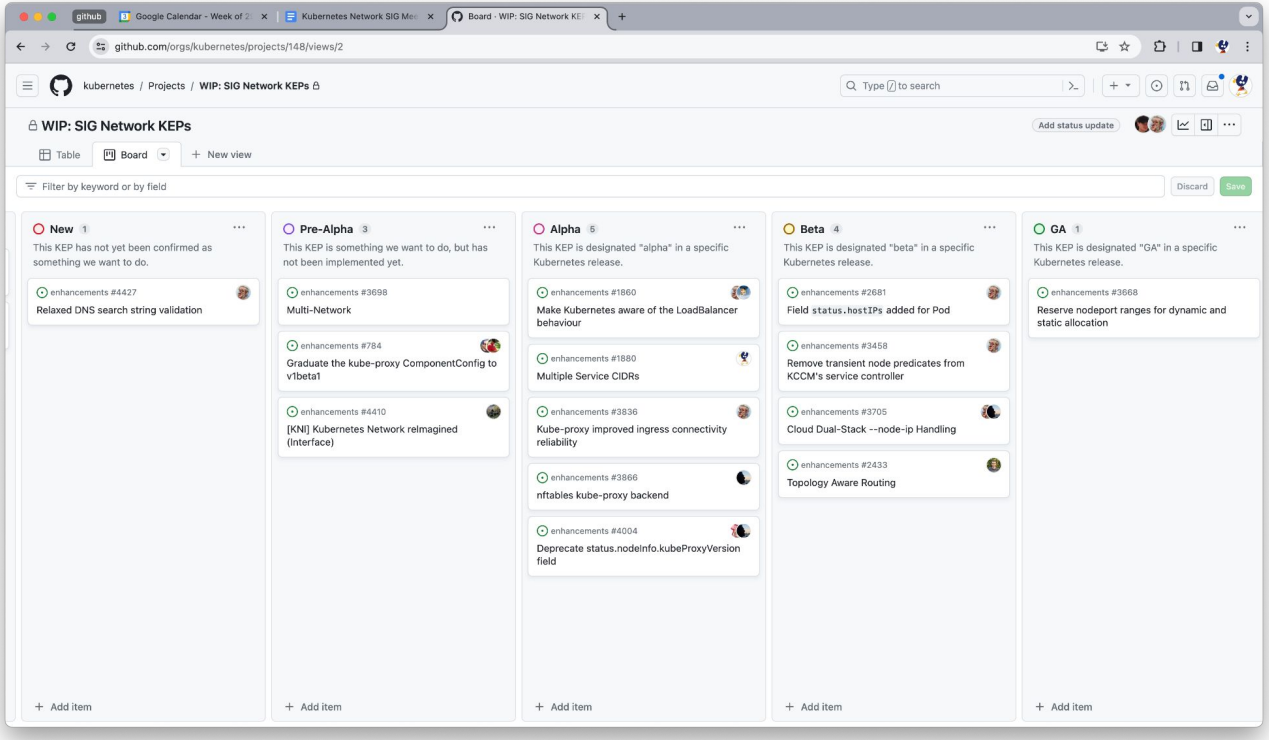



A modern set of APIs for deploying L4 and L7 routing in Kubernetes

Designed to be generic, expressive, extensible, and role-oriented


api
resources
aggregator
admission
exec,
attach, ...
proxy
storage
grpc
External
Load balancer
user
user
ep
svc
api
aggregated
apiserver
ep
svc
crd
webhook
sched
c-m
controlplane node
svc
ep
kubernetes.default.svc
kubelet
pod
pod
worker node
etcd
etcd
etcd
etcd cluster
kubelet
pod
worker node

# What is next


github.com/orgs/kubernetes/projects/148/views/2
kubernetes / Projects / WIP: SIG Network KEPs
WIP: SIG Network KEPs
Table
Board
New view
Filter by keyword or by field
Discard
Save
New 1
This KEP has not yet been confirmed as something we want to do.
enhancements #4427
Relaxed DNS search string validation
Add item
Pre-Alpha 3
This KEP is something we want to do, but has not been implemented yet.
enhancements #3698
Multi-Network
enhancements #784
Graduate the kube-proxy ComponentConfig to v1beta1
enhancements #4410
[KNI] Kubernetes Network reImagined (Interface)
Add item
Alpha 5
This KEP is designated "alpha" in a specific Kubernetes release.
enhancements #1860
Make Kubernetes aware of the LoadBalancer behaviour
enhancements #1880
Multiple Service CIDRs
enhancements #3836
Kube-proxy improved ingress connectivity reliability
enhancements #3866
nftables kube-proxy backend
enhancements #4004
Deprecate status.nodeInfo.kubeProxyVersion field
Add item
Beta 4
This KEP is designated "beta" in a specific Kubernetes release.
enhancements #2681
Field status.hostIPs added for Pod
enhancements #3458
Remove transient node predicates from KCCM's service controller
enhancements #3705
Cloud Dual-Stack --node-ip Handling
enhancements #2433
Topology Aware Routing
Add item
GA 1
This KEP is designated "GA" in a specific Kubernetes release.
enhancements #3668
Reserve nodeport ranges for dynamic and static allocation
Add item