

Managing Ingress Traffic Patterns for Kubernetes Services

USING THE KUBERNETES SERVICE API OBJECT TO EXPOSE WORKLOADS

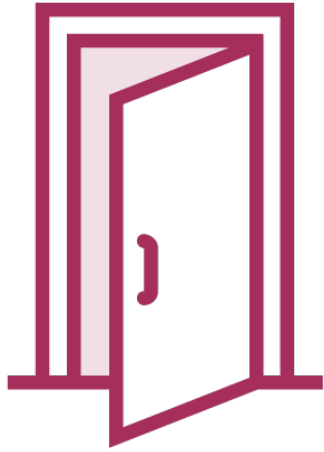


Nigel Brown

@n_brownuk www.windsock.io

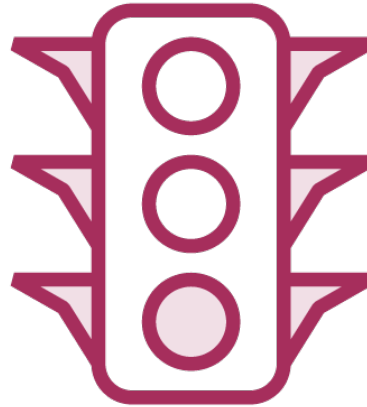


What Is Ingress?



Enabling Traffic

Opening the cluster to receive external traffic



Traffic Routing

Defining traffic routes to backend services



Traffic Reliability

Ensuring reliable, secure communication



Module Outline



Networking in Kubernetes

Abstracting pod workloads as services

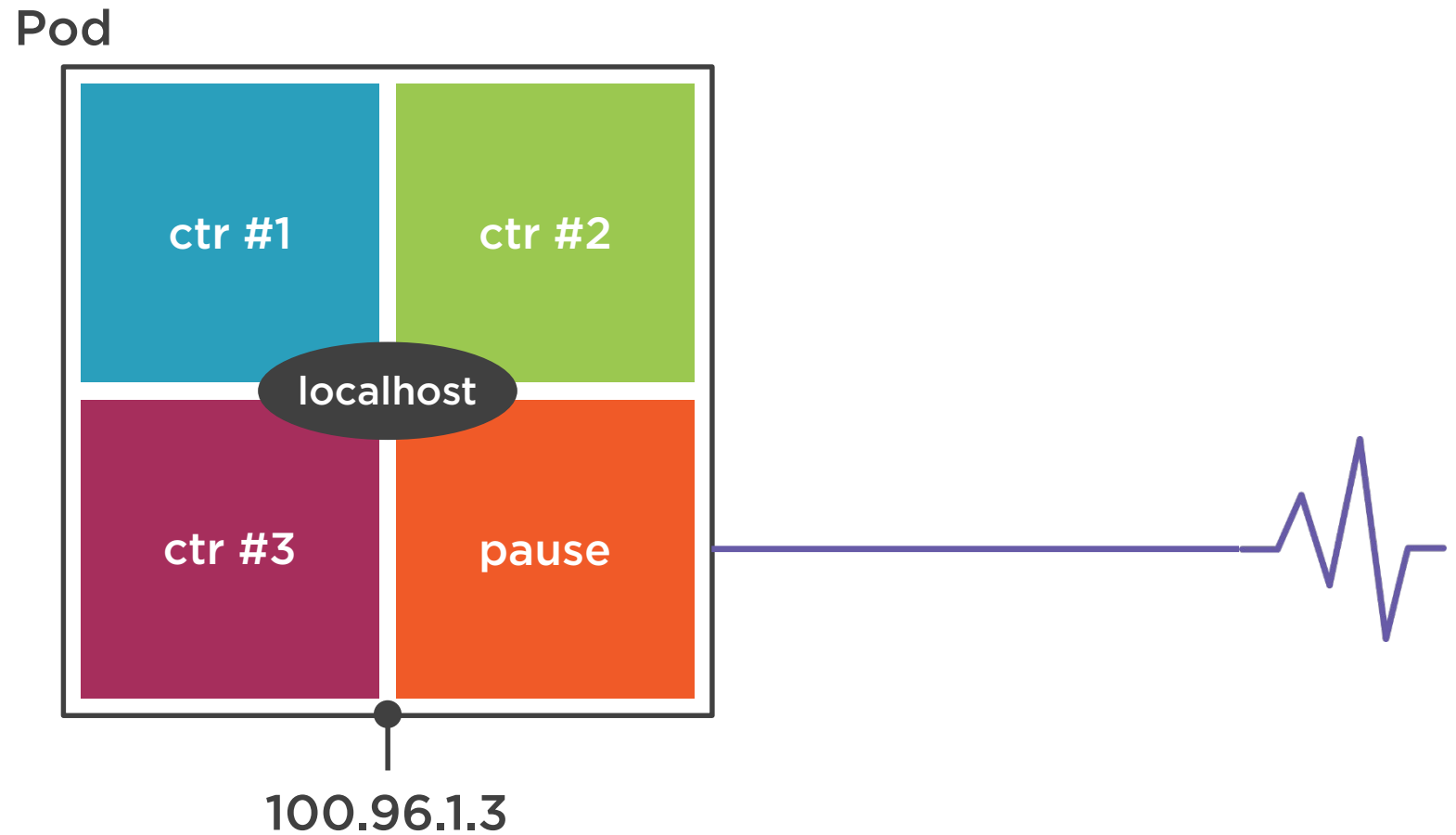
Using the Service API to manage ingress

Exposing workloads with the Service API

The limitations of the Service API

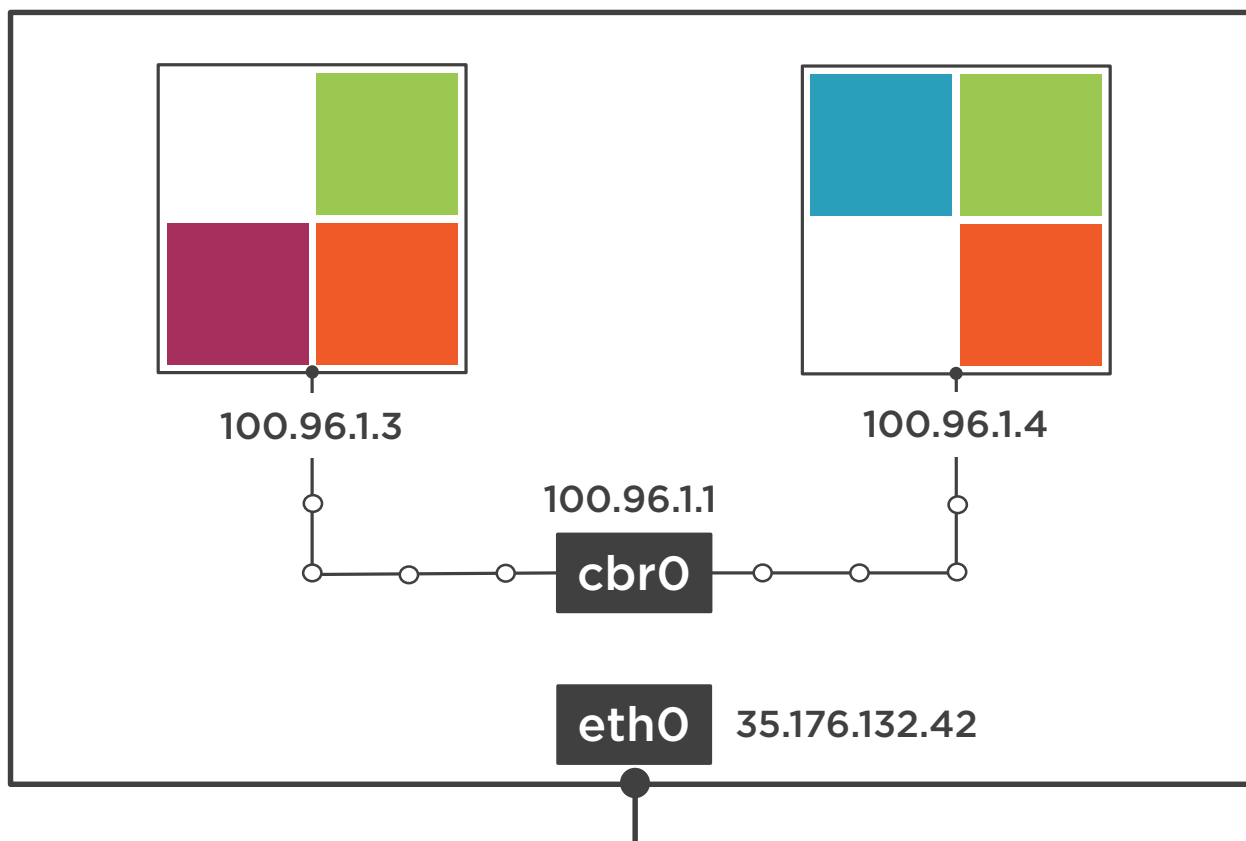


Container Networking



Pod Networking - Same Node

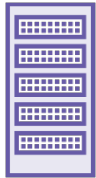
Node



Kubernetes Networking Rules



All pods can communicate with all other pods without Network Address Translation (NAT)



All nodes can communicate with all pods (and vice-versa) without NAT

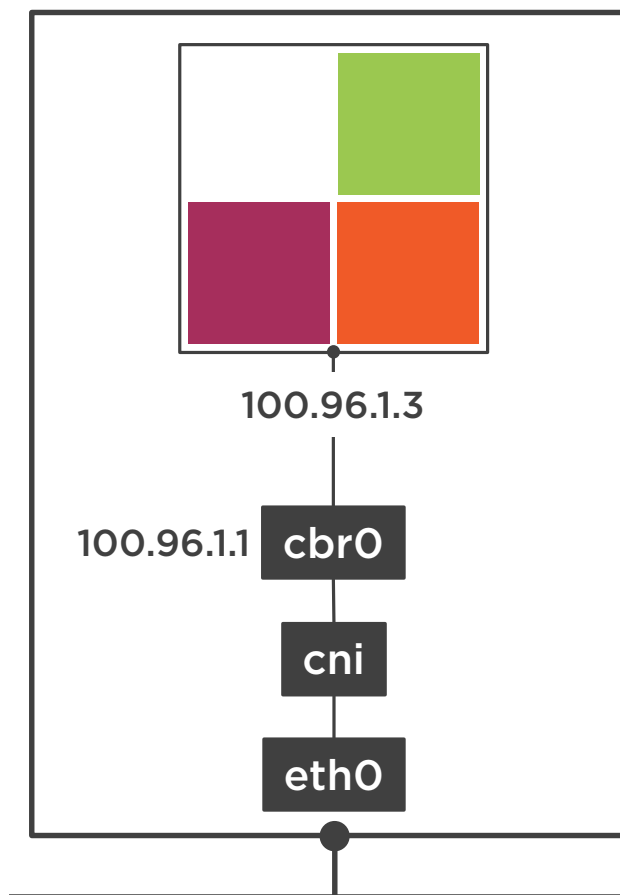


The IP address that a pod sees itself as, is the same IP address that others see it as

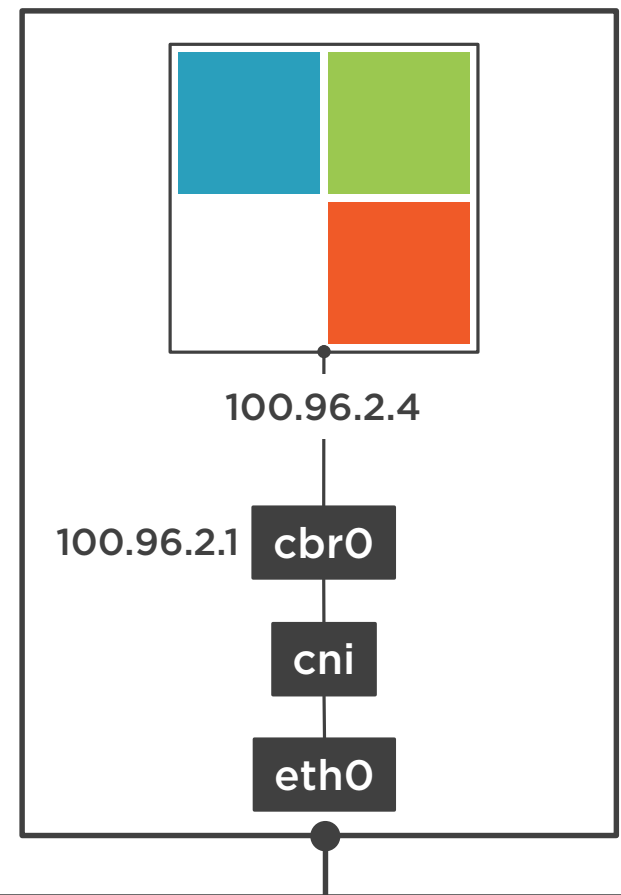


Pod Networking - Across Nodes

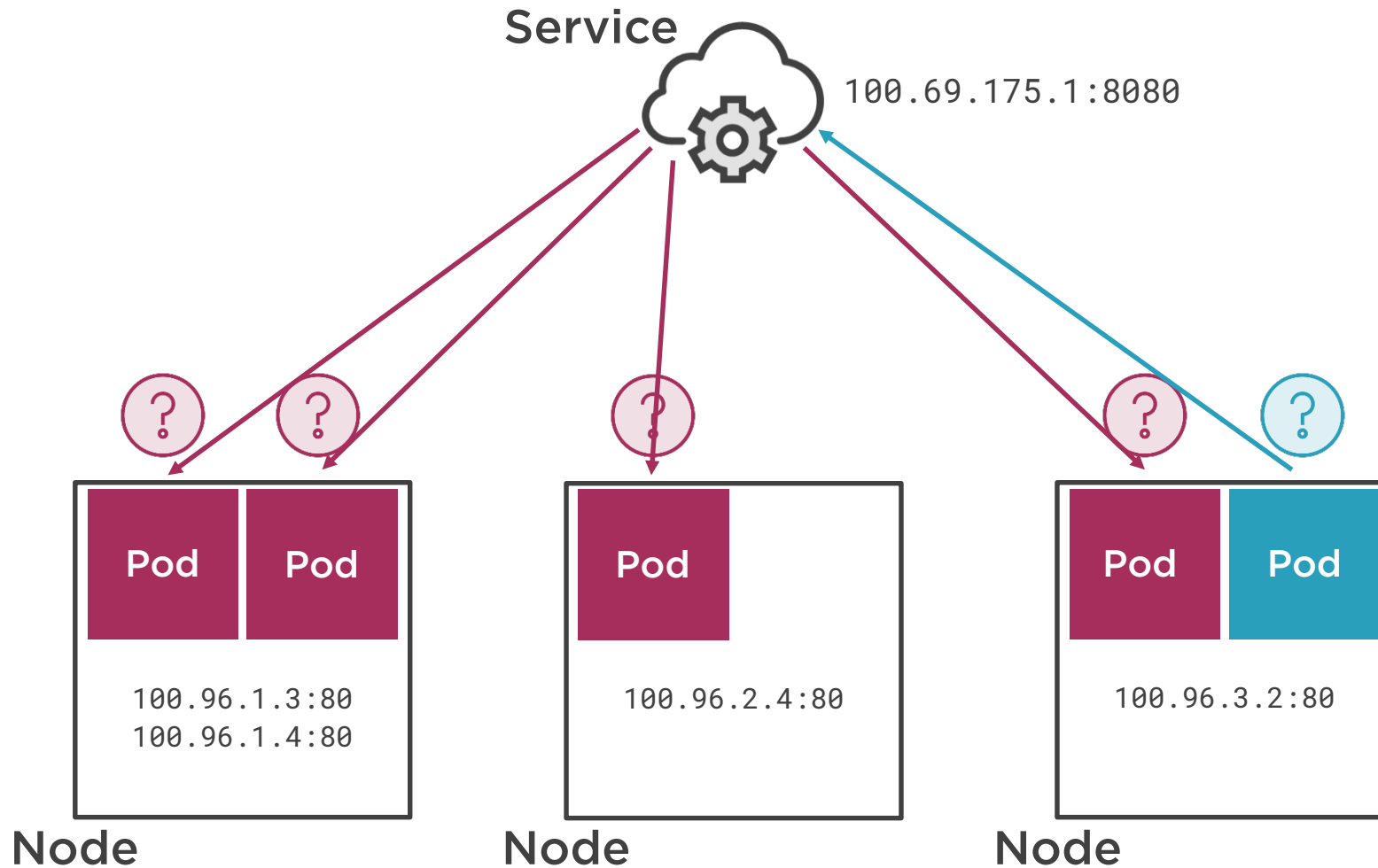
Node



Node



Using the Service Abstraction




```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
    name: nginx
spec:
  clusterIP: 100.69.175.1
  ports:
    - port: 8080
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
type: ClusterIP
```

- ◀ The service's virtual IP address
- ◀ Ports exposed by the service
- ◀ Labels used to select target pods
- ◀ Type defines how service is exposed



```
$ kubectl get endpoints nginx
```

NAME	ENDPOINTS	AGE
nginx	100.96.1.3:80,100.96.1.4:80,100.96.2.4:80	3h

Service Endpoints

Endpoints API objects hold information on each of a service's pods



```
$ kubectl run -it --rm nwutils --restart=Never \
    --image=nbrown/nwutils -- nslookup nginx
Server:      10.96.0.10
Address: 10.96.0.10#53

Name:  nginx.default.svc.cluster.local
Address: 100.69.175.1
```

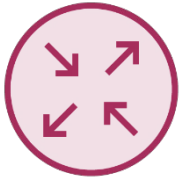
Service Discovery

Services can be found using environment variables, but ...

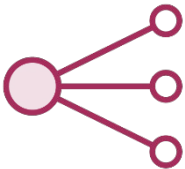
... it's better to rely on an in-cluster DNS for looking up services



Proxying Traffic to Service Endpoints



The kube-proxy is the cluster component that enables traffic routing



It load balances traffic between pods using iptables or IP Virtual Server (IPVS)



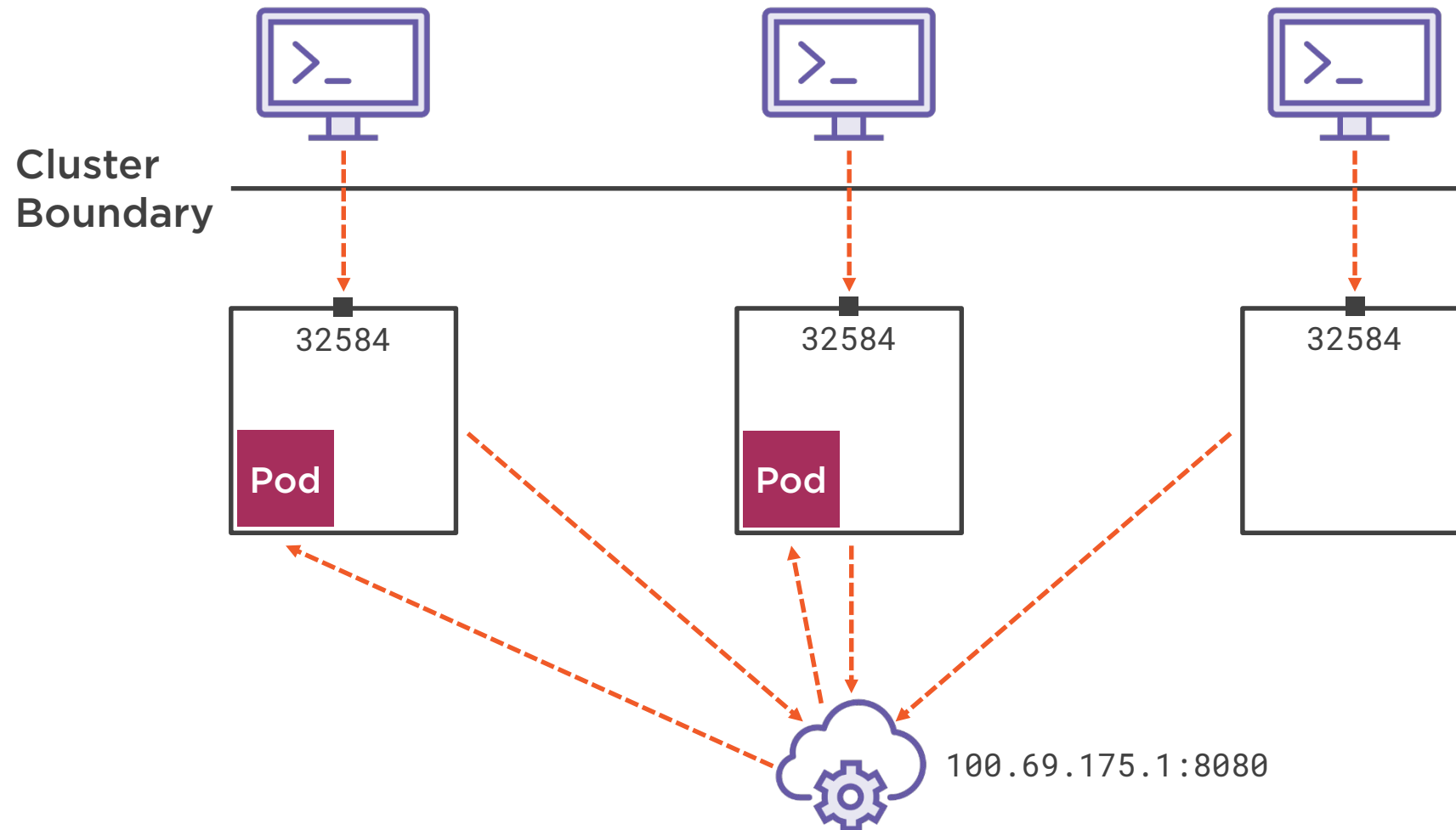
It watches for state change, and re-defines the proxy configuration accordingly



By default, services are
isolated from clients
external to the cluster



NodePort Type



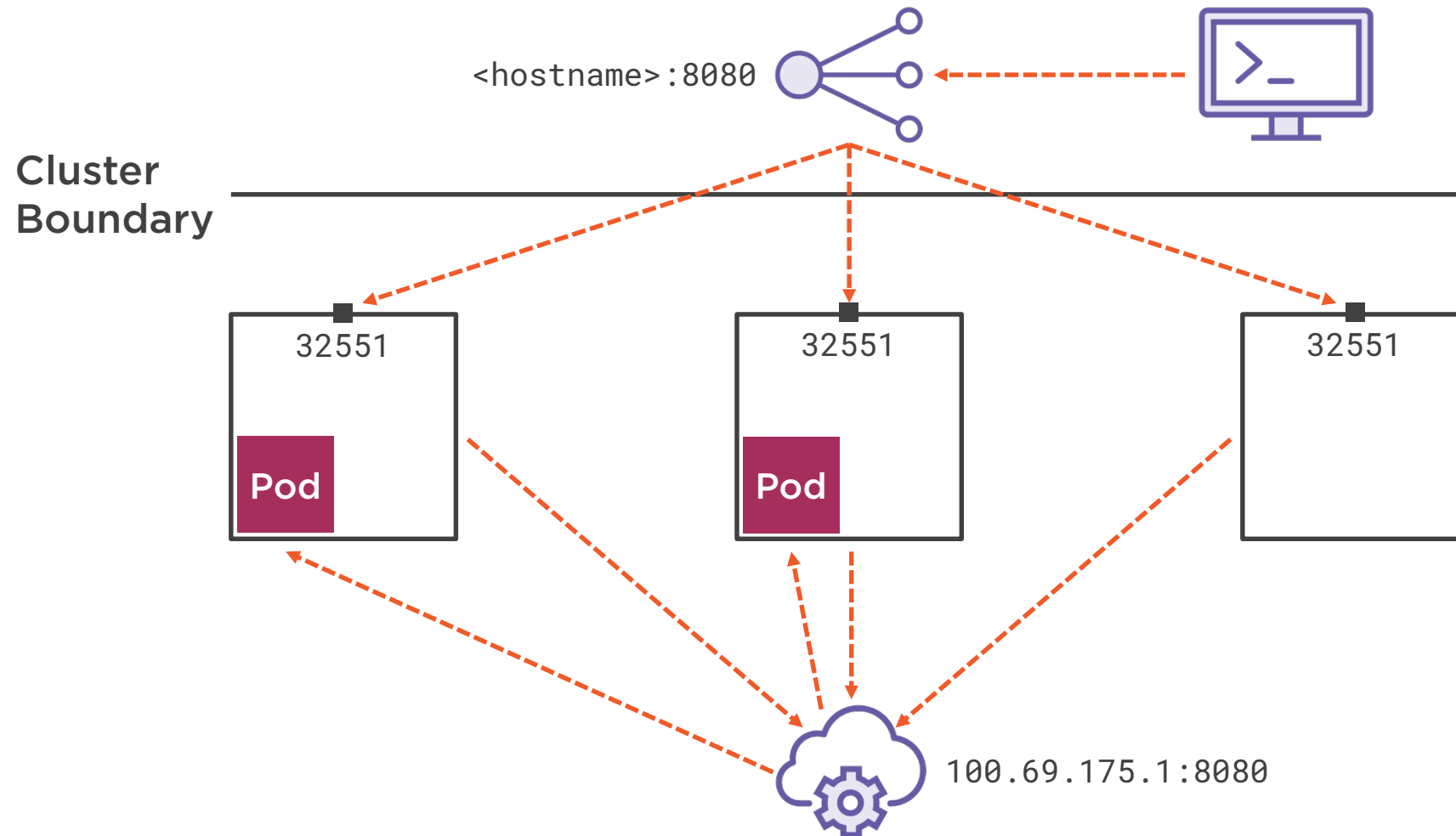
```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  clusterIP: 100.69.175.1
  externalTrafficPolicy: Cluster
  ports:
    - nodePort: 32584
      port: 8080
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
type: NodePort
```

- ◀ Service retains clusterIP
- ◀ Policy for traffic routing
- ◀ Node port for external traffic

- ◀ Service type set to NodePort



LoadBalancer Type




```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  clusterIP: 100.69.175.1
  externalTrafficPolicy: Cluster
  ports:
    - nodePort: 32551
      port: 8080
      protocol: TCP
      targetPort: 80
  selector:
    app: nginx
  type: LoadBalancer
  ingress:
    - hostname: ...
```

- ◀ Service retains clusterIP
- ◀ Policy for traffic routing
- ◀ Node port used by external load balancer
- ◀ Service type is LoadBalancer



Demo



How to expose a workload using a Service API object

Workload will be exposed in-cluster, before being configured for ingress

The Kubernetes cluster runs on the AWS cloud platform



Limitations of the Service API



Manual configuration of load balancer when using NodePort type



Potential latency due to the network hops introduced by kube-proxy



A load balancer per service can quickly escalate operational costs



The Service API cannot cater for advanced ingress traffic patterns



Module Summary



The Kubernetes Service API is built on top of a flat networking model

The Service API allows for abstracting replicated workloads

The Service API is able to cater for basic ingress requirements

