

Kubernetes.
Сетевое взаимодействие.
Хранение данных.

план

- Сетевое взаимодействие в Kubernetes
- Хранение данных в Kubernetes

Сетевое взаимодействие в Kubernetes

Pods

- Смертны
- Динамически создаются и удаляются
- IP-адреса появляются и исчезают вместе с Pod'ами
- Несколько Pod'ов могут выполнять одну и ту же функцию

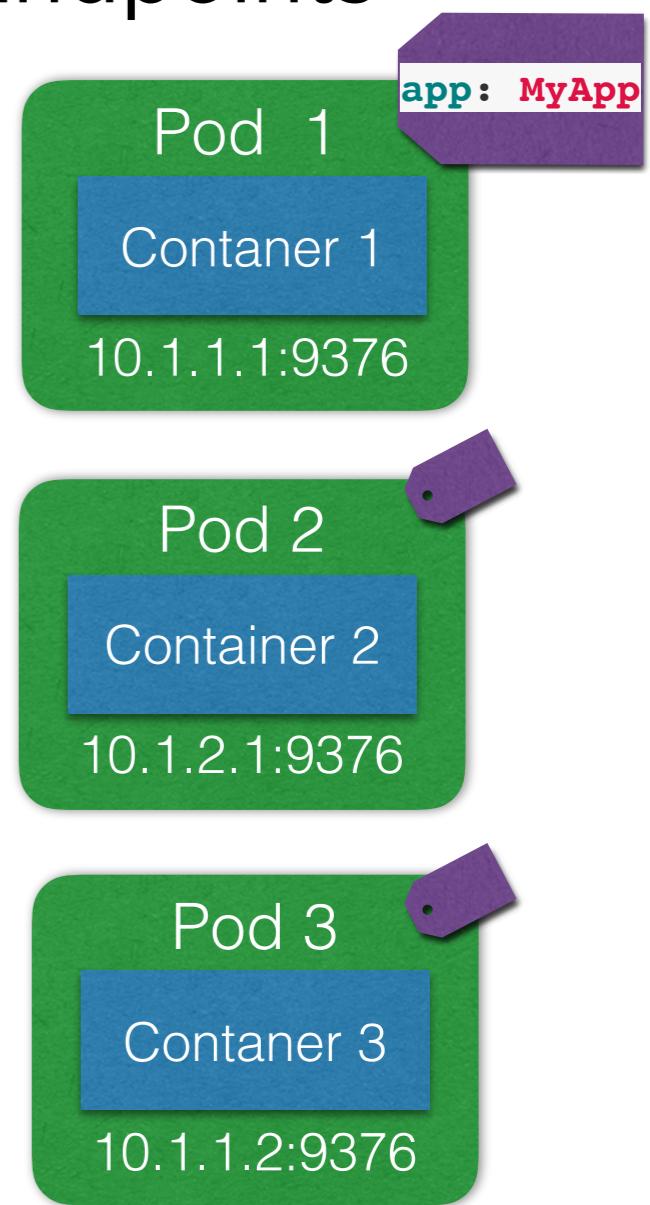
Service

- Абстракция, описывающая набор Pod'ов и конфигурацию доступа к ним
- Позволяет отвязаться от использования конкретных Pod'ов

Service with selector

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

Endpoints



Service without selector

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

```
kind: Endpoints
apiVersion: v1
metadata:
  name: my-service
subsets:
  - addresses:
      - ip: 1.2.3.4
    ports:
      - port: 9376
```

ExternalName service

my-service.prod.svc.cluster.local => my.database.example.com

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
  namespace: prod
spec:
  type: ExternalName
  externalName: my.database.example.com
```

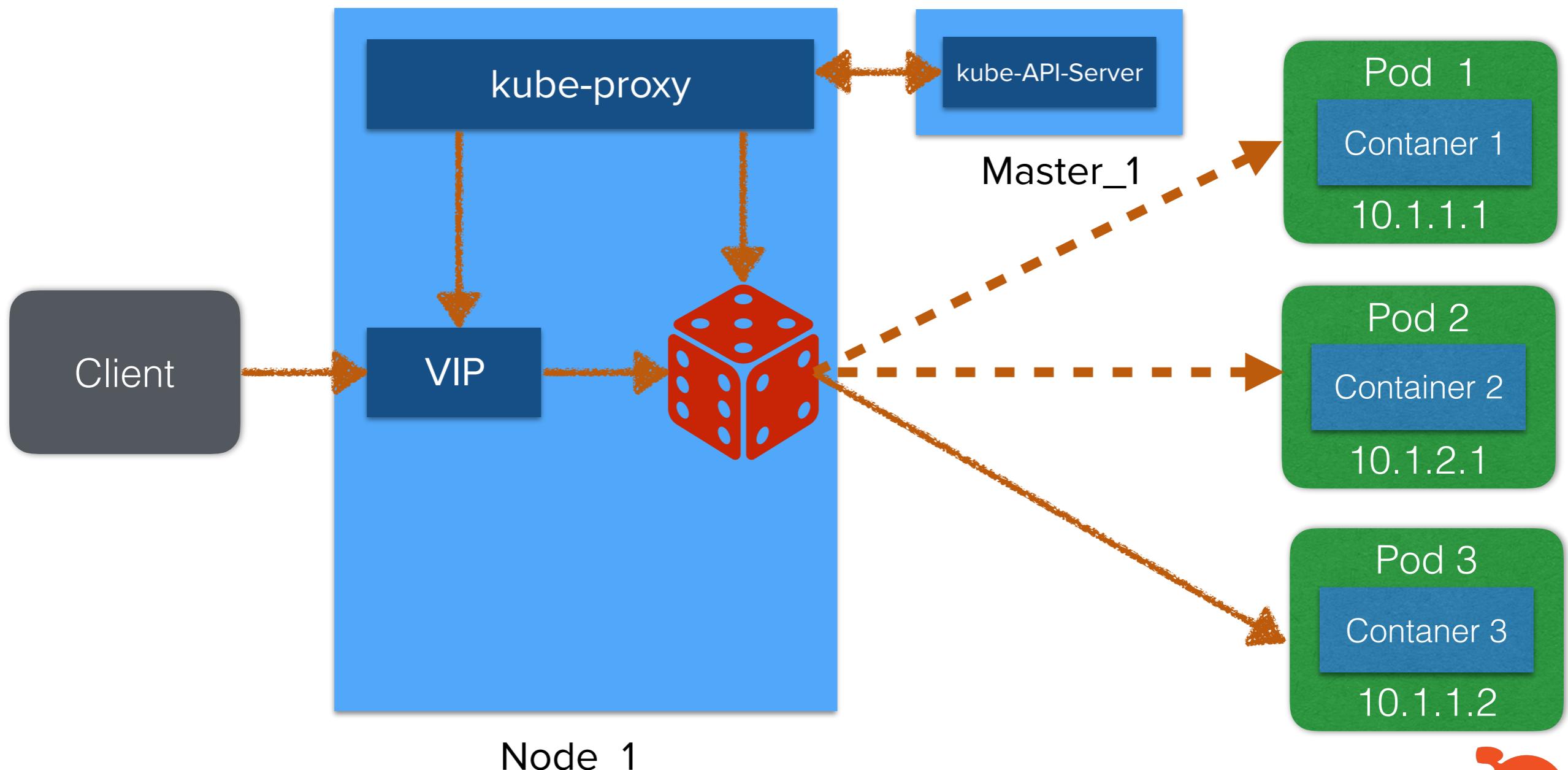
Headless Service

- Service без Cluster IP
- DNS возвращает адреса Pod'ов (Service with Selector)
- Возвращает CNAME запись (для ExternalName Service)
- Возвращает адреса всех одноименных сервису Endpoint'ов (во всех остальных случаях)

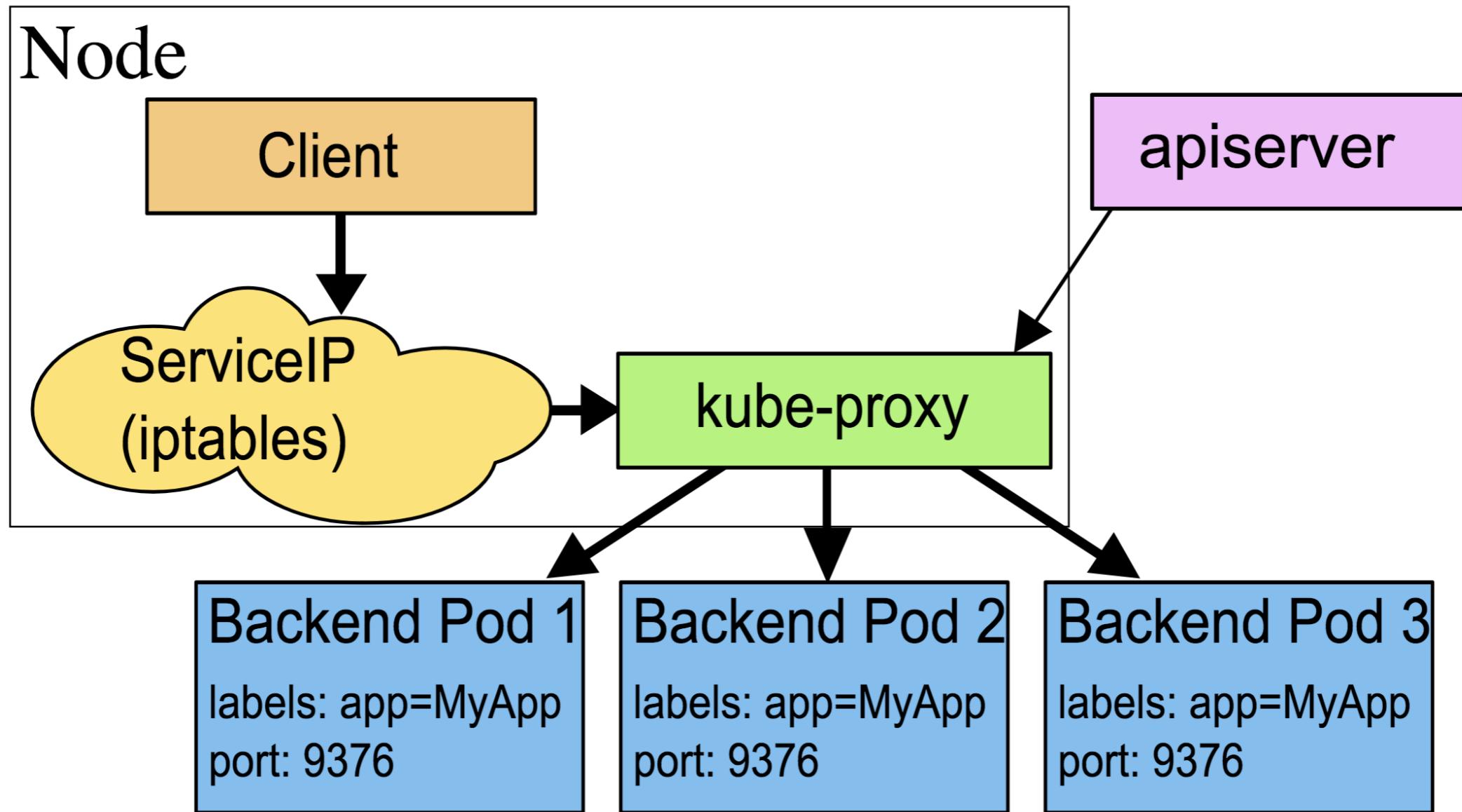
kube-proxy

- Проксирует TCP и UPD (не HTTP)
- Используется только, для работы с сервисами
- Обеспечивает внутреннюю балансировку
- Используется для внутренней коммуникации

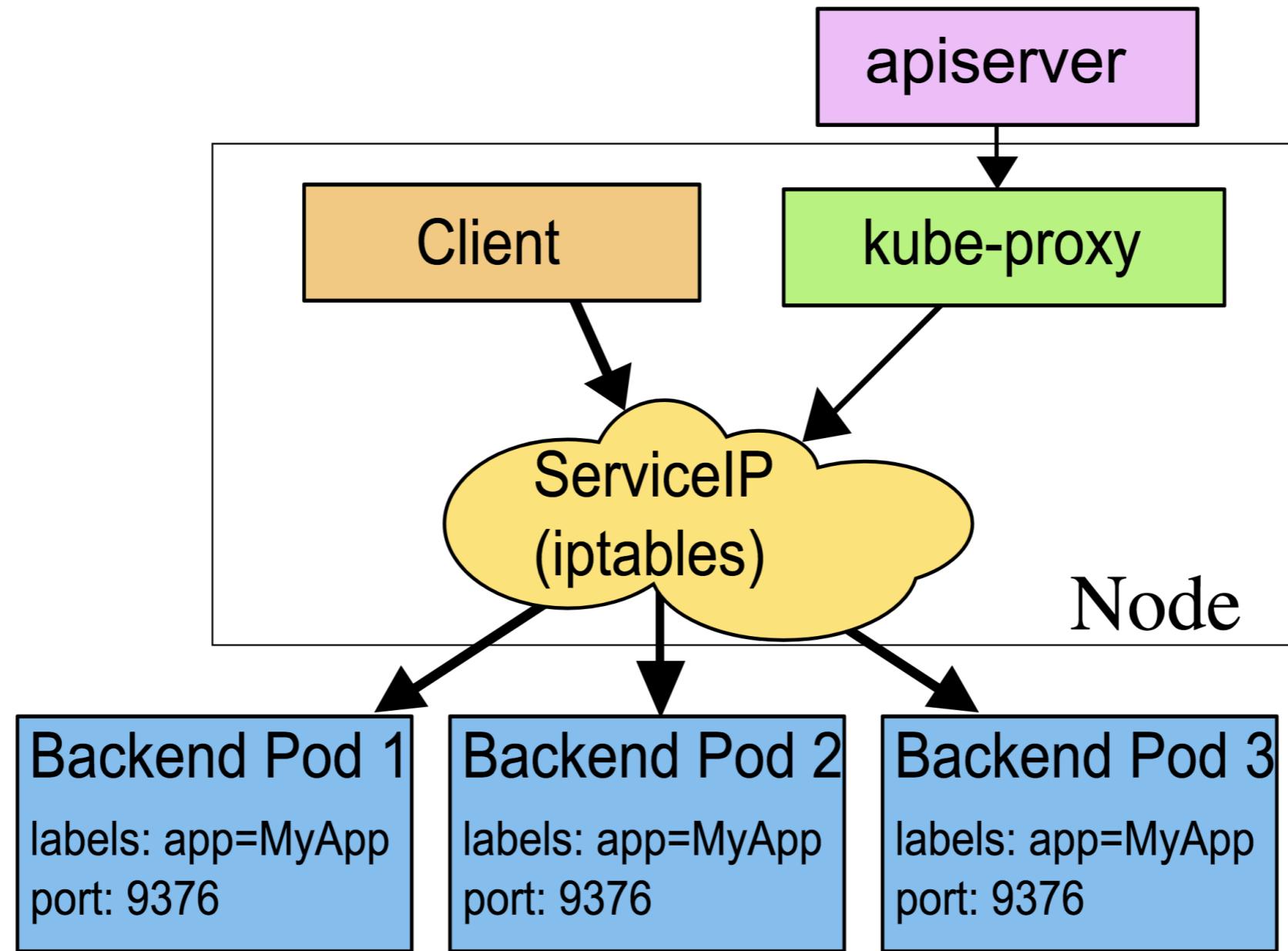
kube-proxy



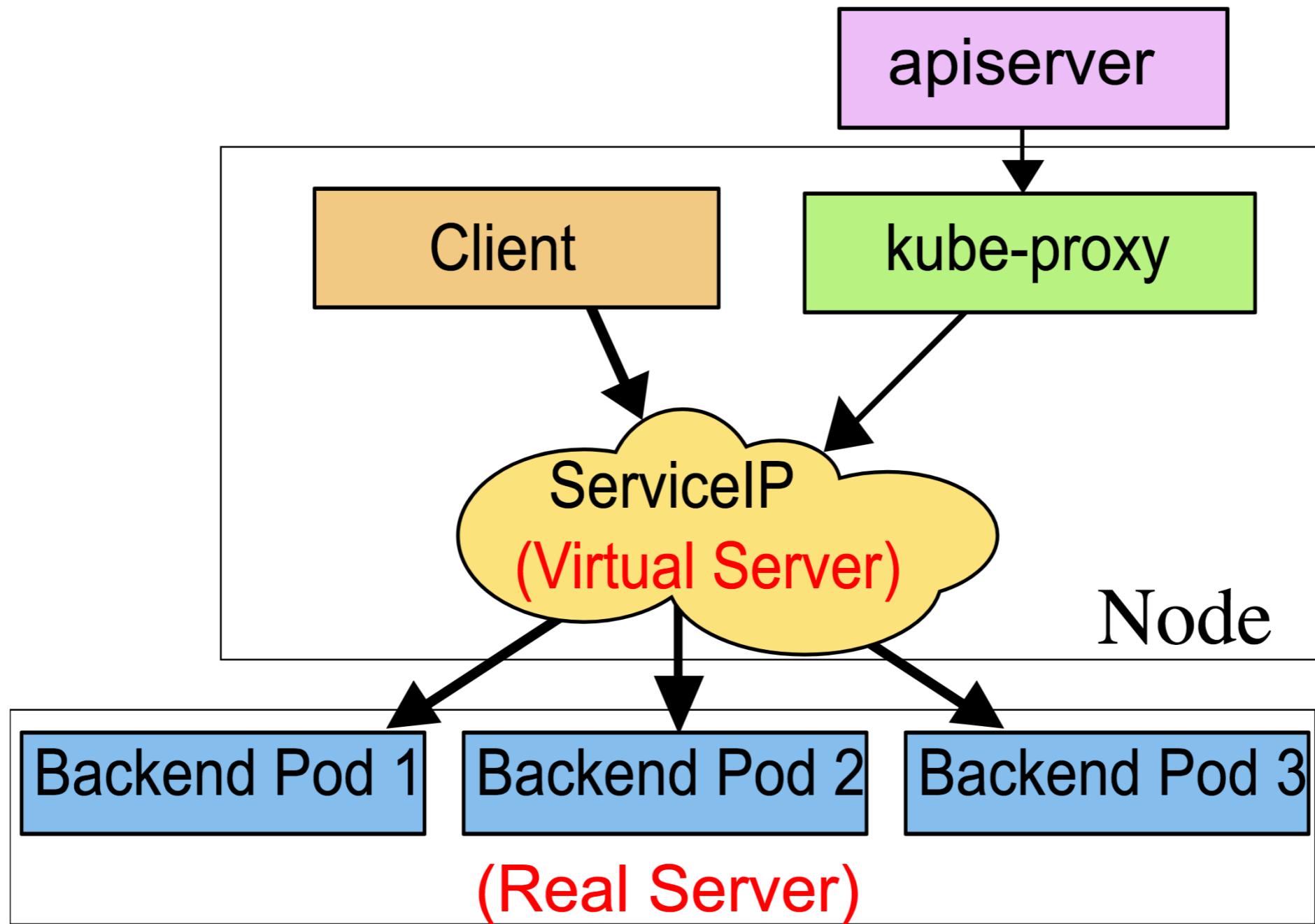
kube-proxy: Userspace



kube-proxy: iptables

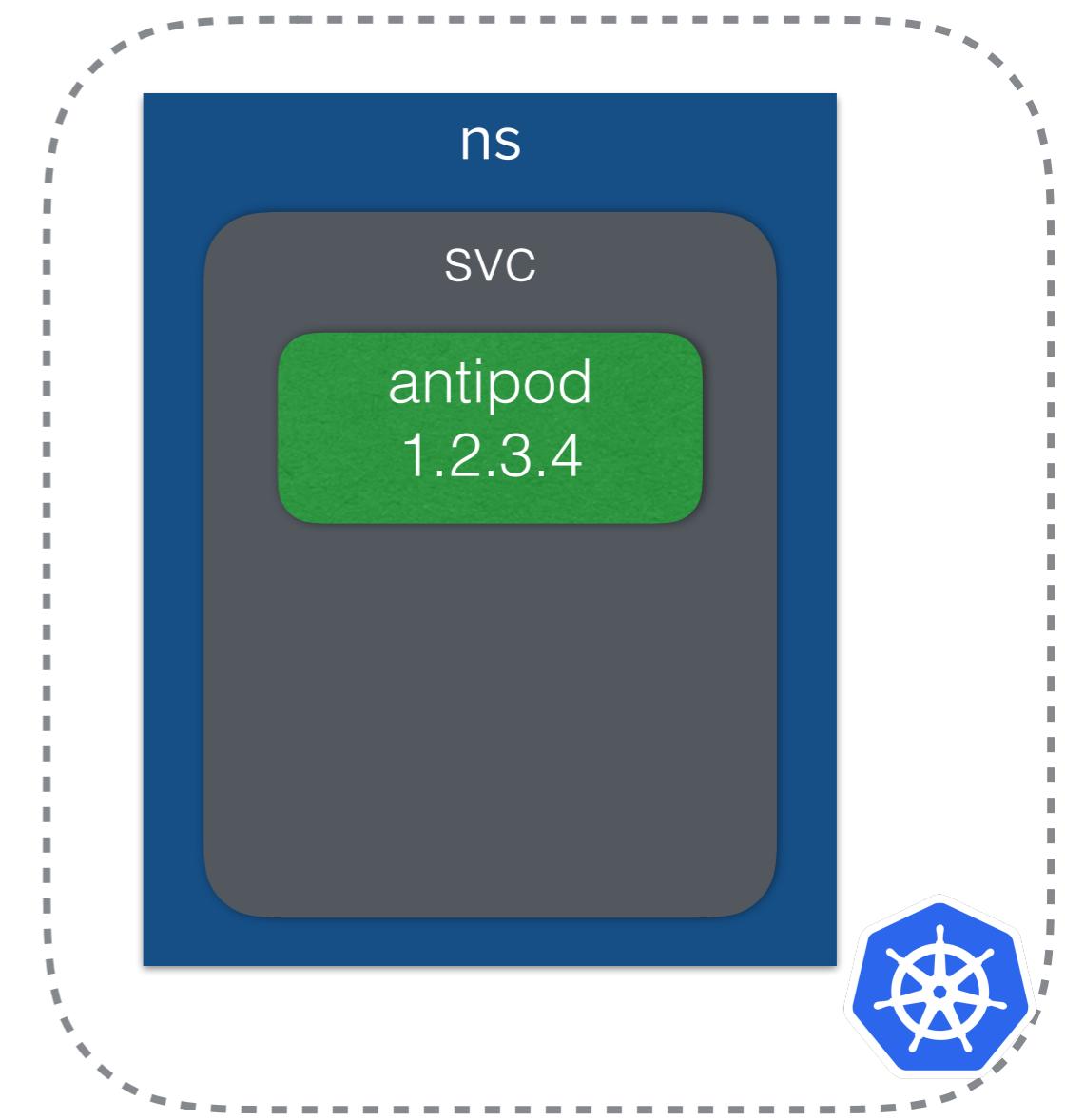


kube-proxy: IPVS

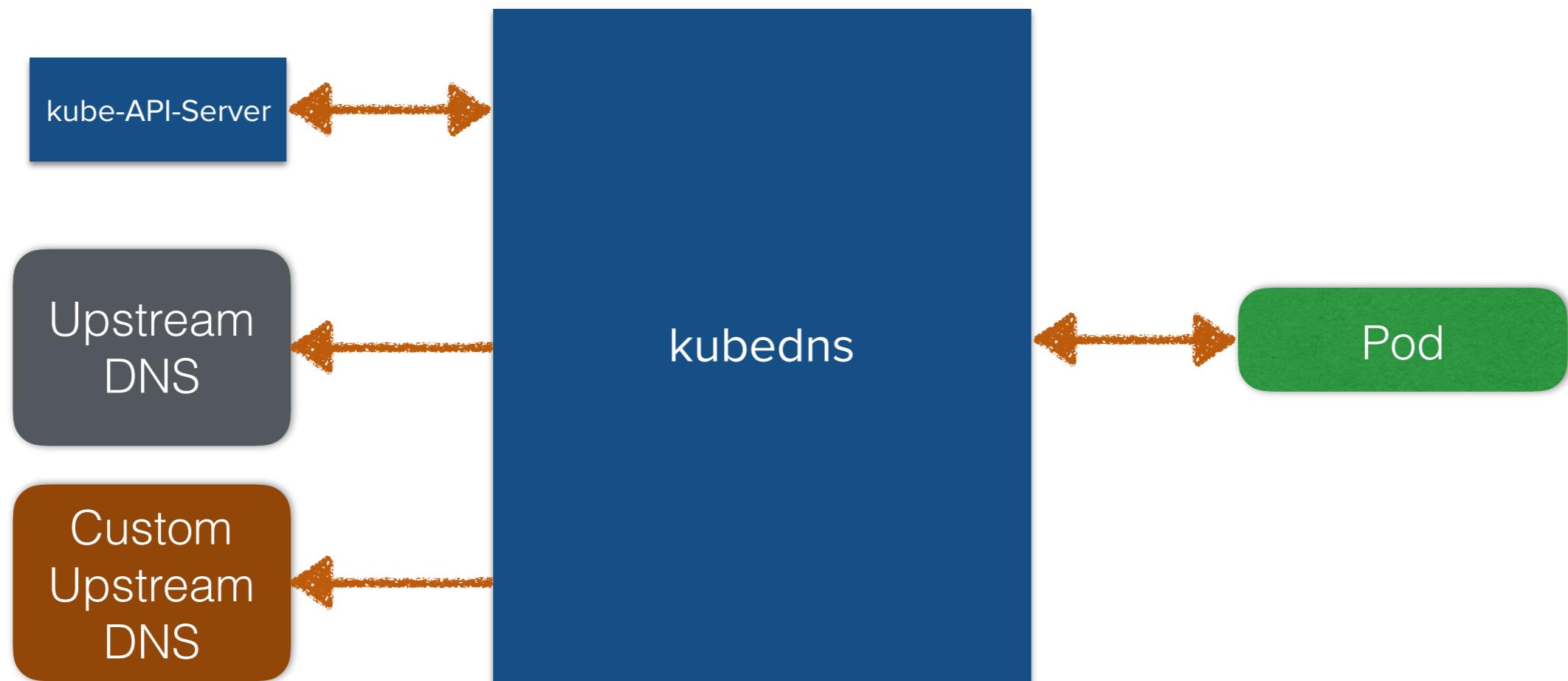


kubedns

- Namespace
 - SVC
- Cluster
 - SVC.ns
 - FQDN
 - svc.ns.svc.cluster.local
 - Pod FQDN
 - 1-2-3-4.ns.pod.cluster.local



kubedns



kubedns config

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
data:
  stubDomains: |
    {"acme.local": ["1.2.3.4"]}
  upstreamNameservers: |
    ["8.8.8.8", "8.8.4.4"].
```

dnsPolicy

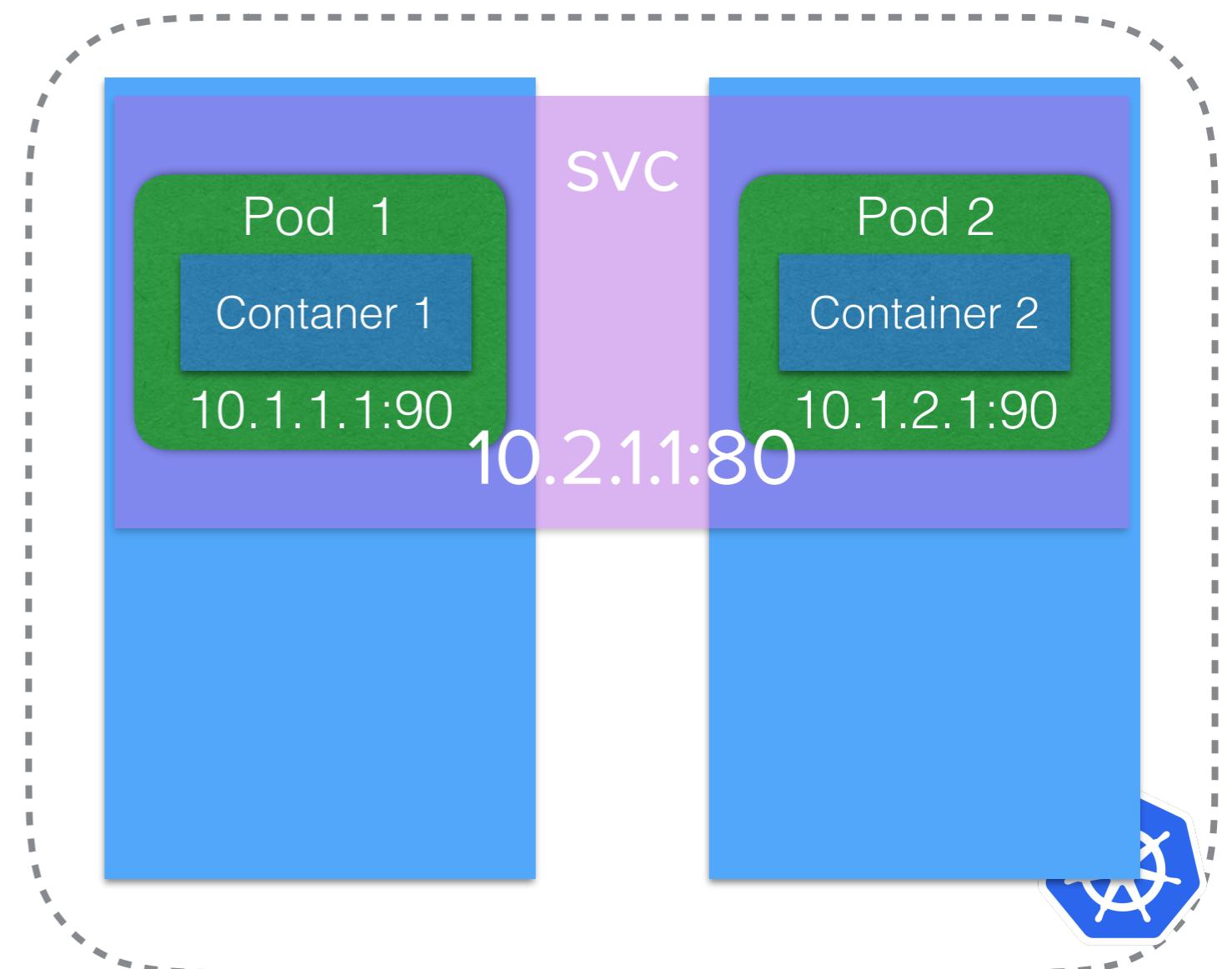
- Default - разрешение имен наследуется с хоста на котором запущен Pod.
Нельзя использовать upstreamNameservers и stubDomains
- ClusterFirst - разрешение имен происходит через kubedns и upstream сервера

Publishing Service

- ClusterIP - только внутри кластера
- NodePort - на порты нод кластера
- LoadBalancer - на внешнем сервисе балансировки
- ExternalIP - указатель на IP-адрес, по которому доступна одни или несколько Node кластера

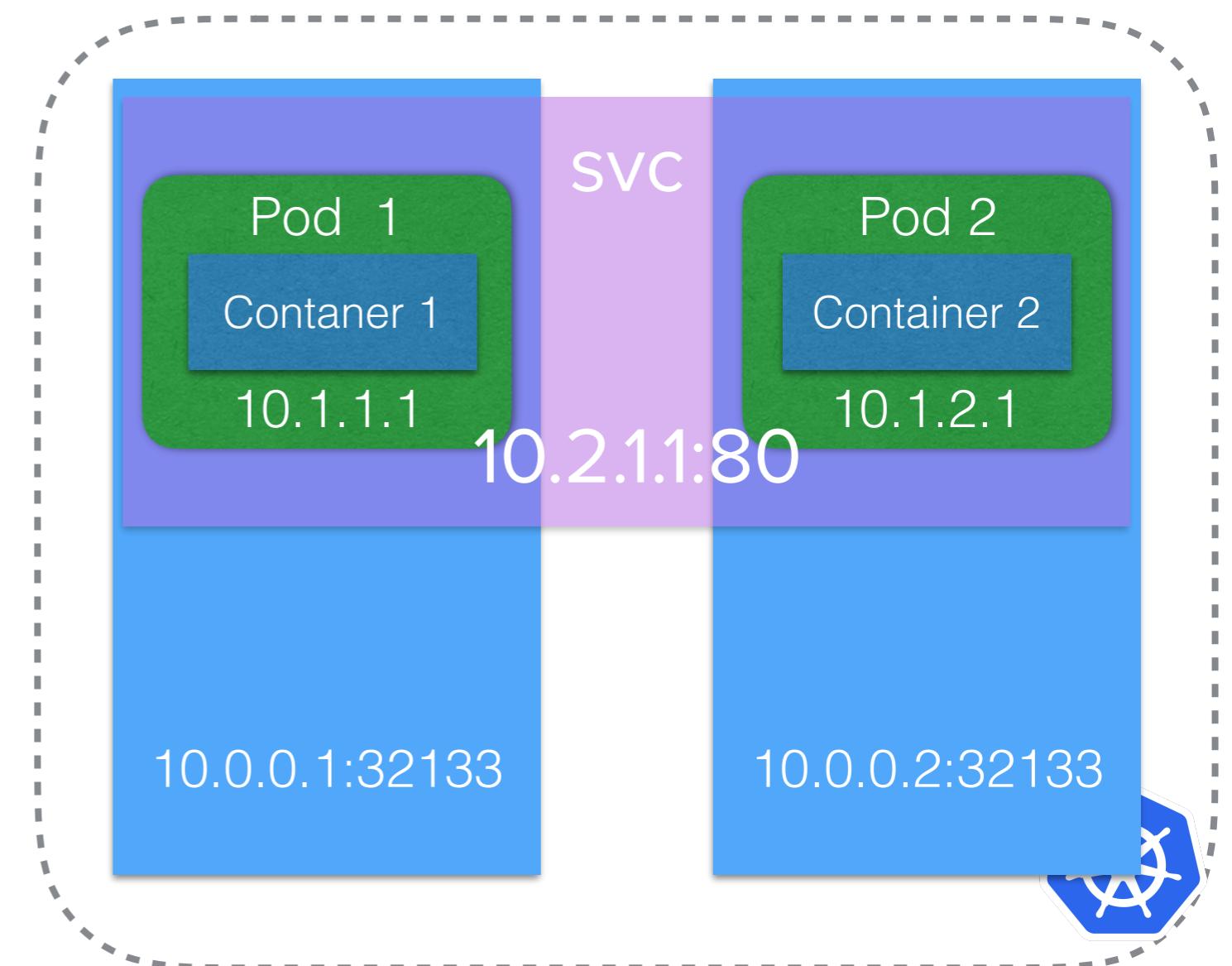
ClusterIP

```
kind: Service
apiVersion: v1
metadata:
  name: svc
spec:
  type: ClusterIP
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 90
```



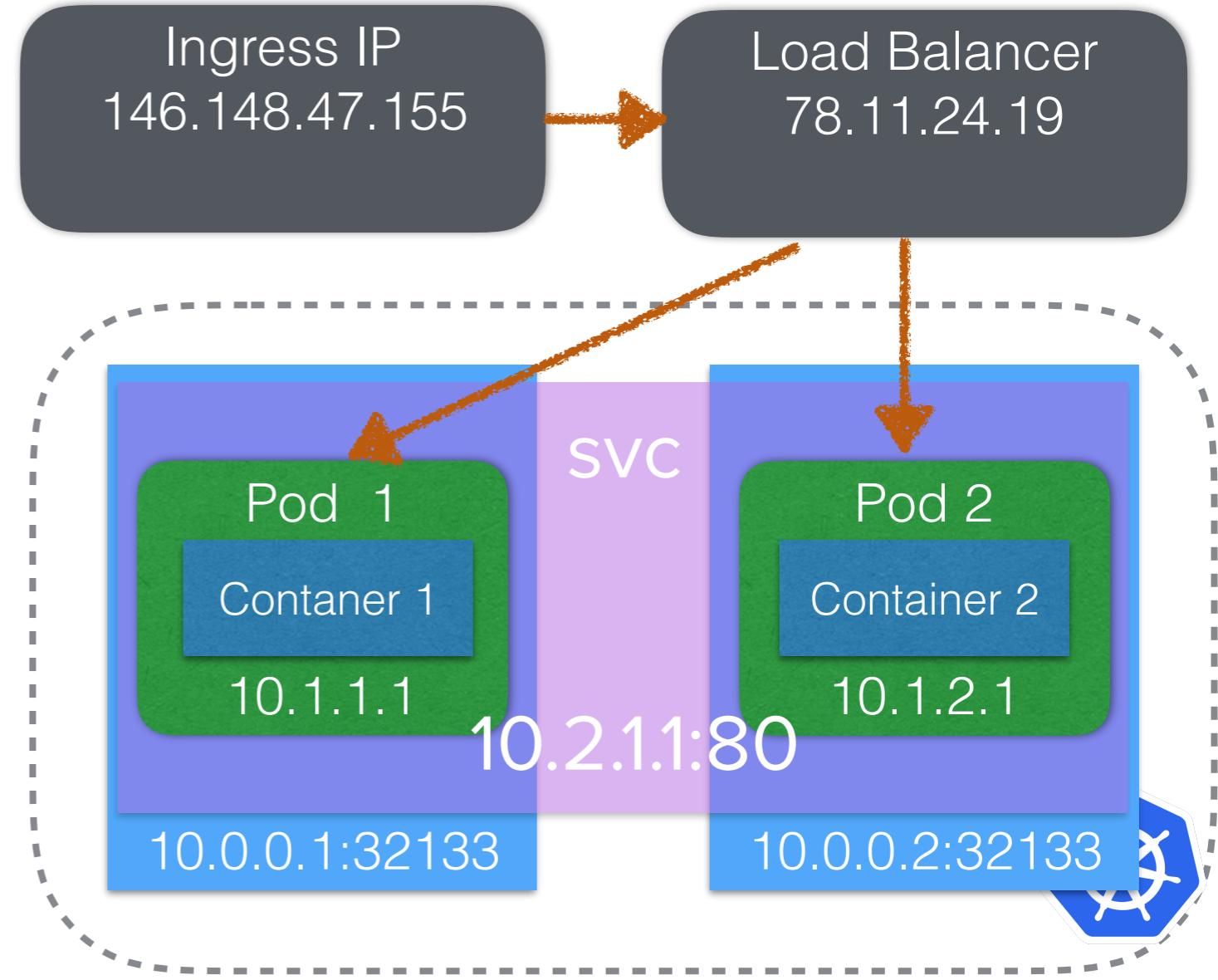
NodePort

```
kind: Service
apiVersion: v1
metadata:
  name: svc
spec:
  type: NodePort
  selector:
    app: MyApp
  ports:
  - port: 80
```



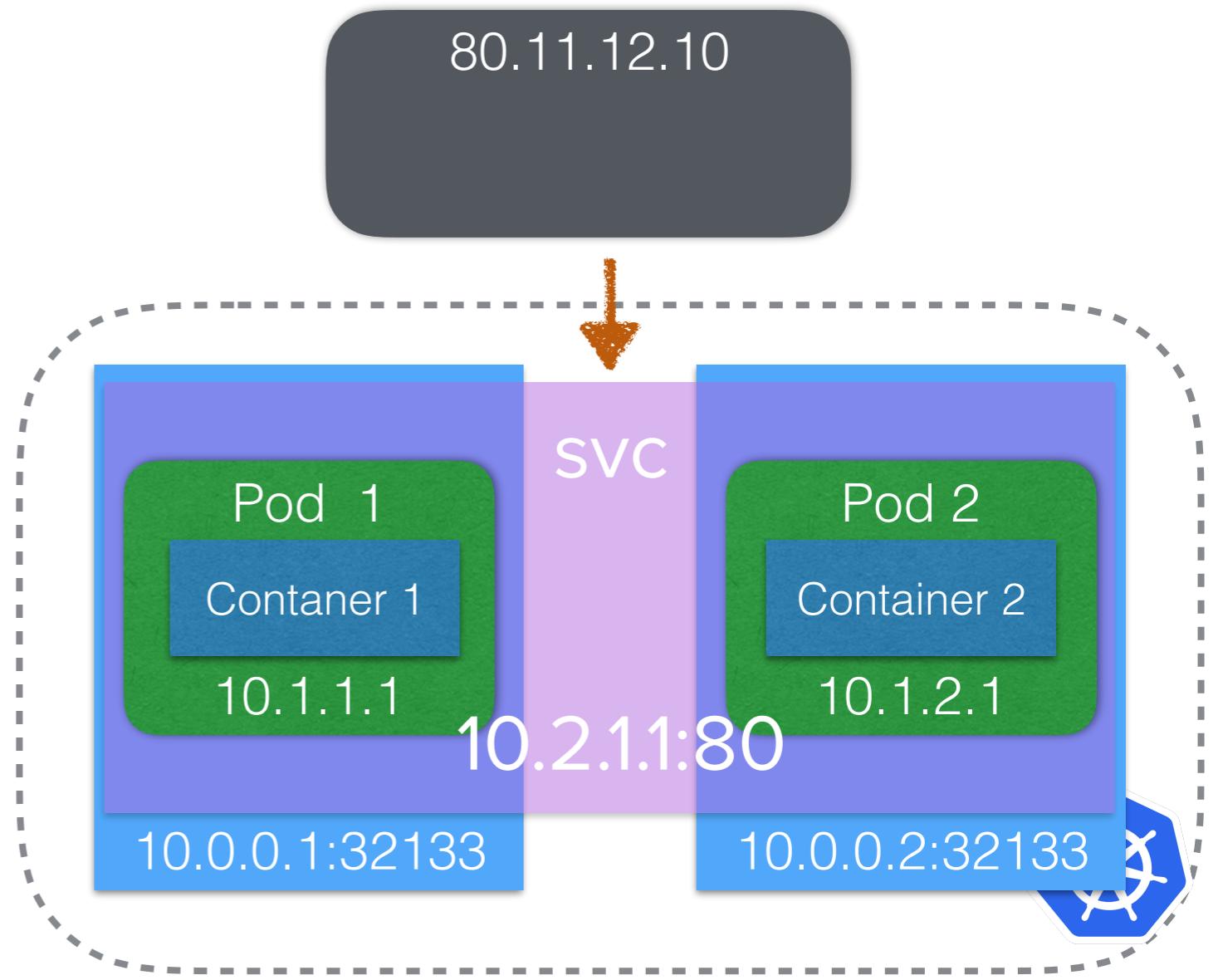
LoadBalancer

```
kind: Service
apiVersion: v1
metadata:
  name: svc
spec:
  type: LoadBalancer
  selector:
    app: MyApp
  ports:
  - port: 80
    clusterIP: 10.2.1.1
    loadBalancerIP: 78.11.24.19
status:
  loadBalancer:
    ingress:
    - ip: 146.148.47.155
```



ExternalIP

```
kind: Service
apiVersion: v1
metadata:
  name: svc
spec:
  selector:
    app: MyApp
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 9376
  externalIPs:
  - 80.11.12.10
```

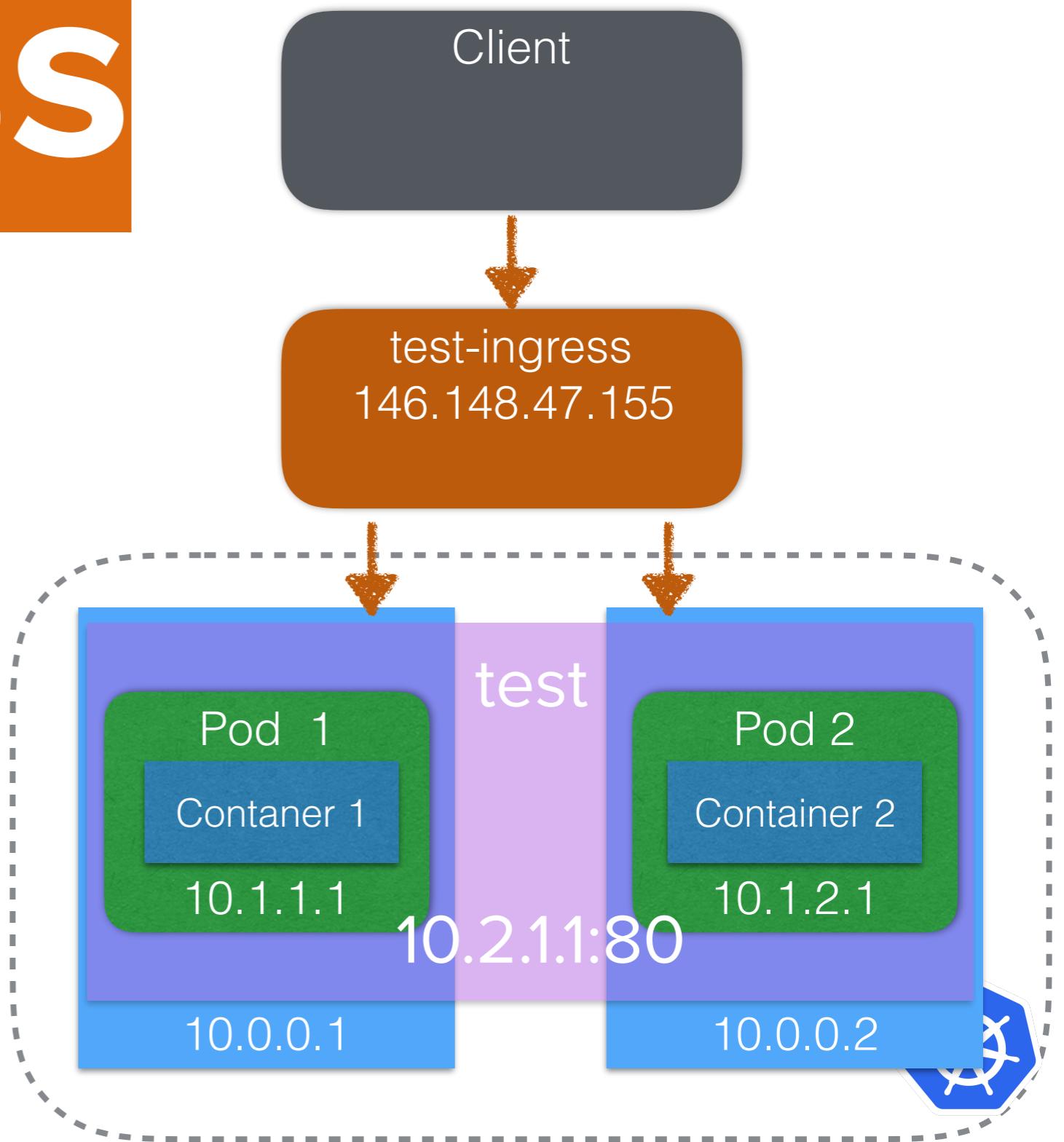


Ingress

- Ingress - объект управляющий внешним доступом к сервисам внутри кластера. Обеспечивает:
 - Балансировку нагрузки
 - Терминацию SSL трафика
 - Name based virtual hosting
- Работает на L3/L4 уровнях

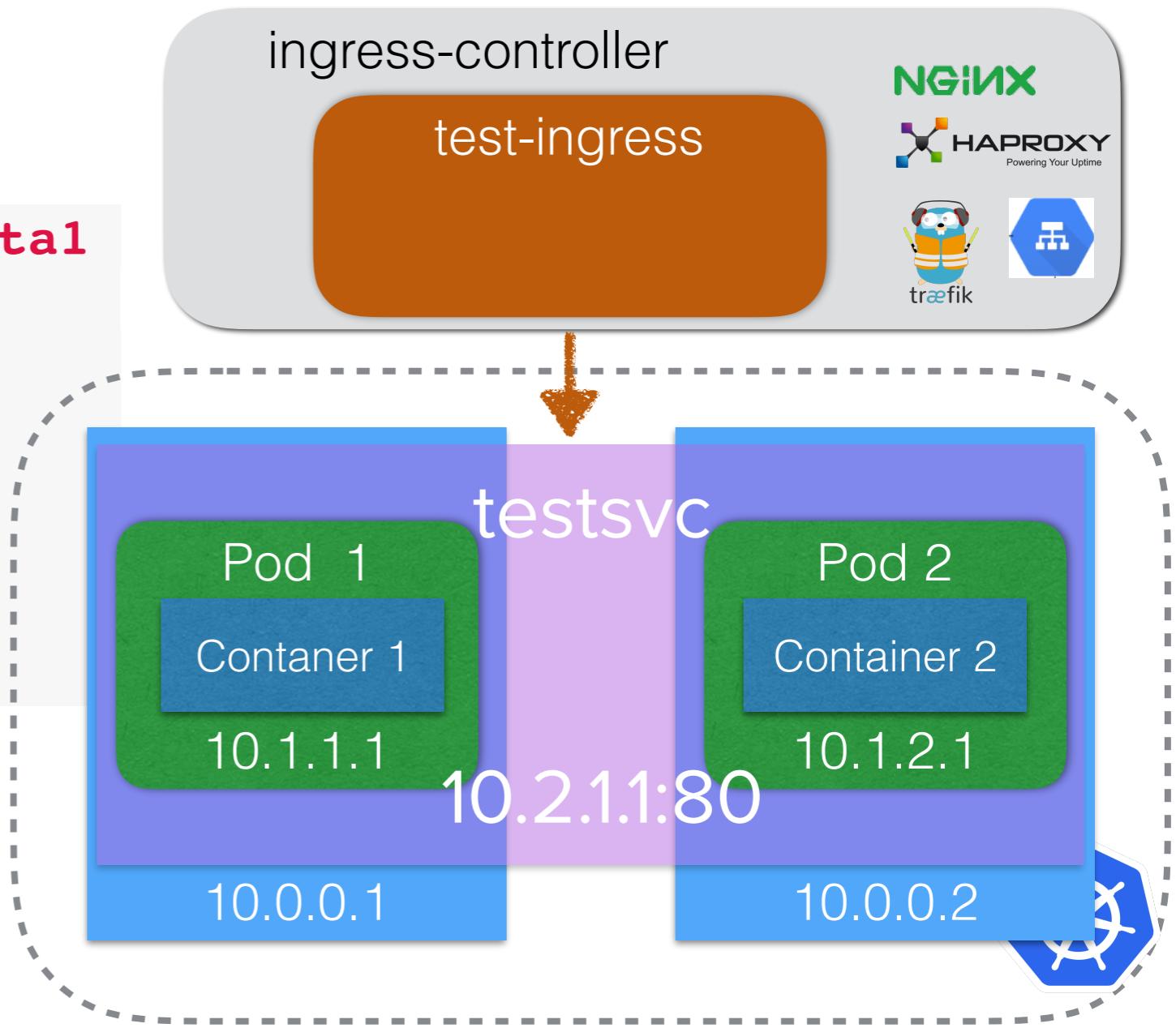
Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    ingress.kubernetes.io/
rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```



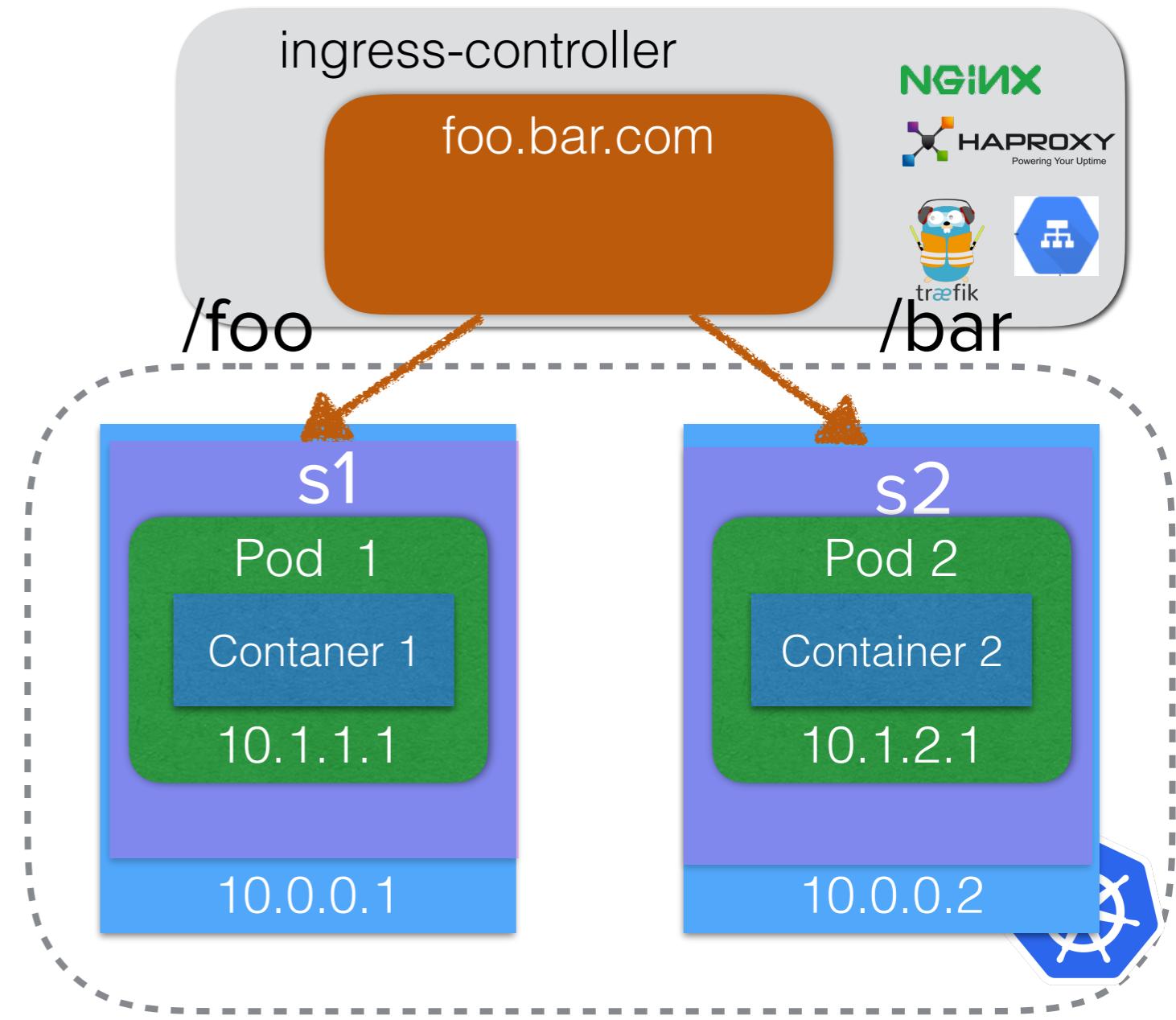
Single Service Ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  backend:
    serviceName: testsvc
    servicePort: 80
```



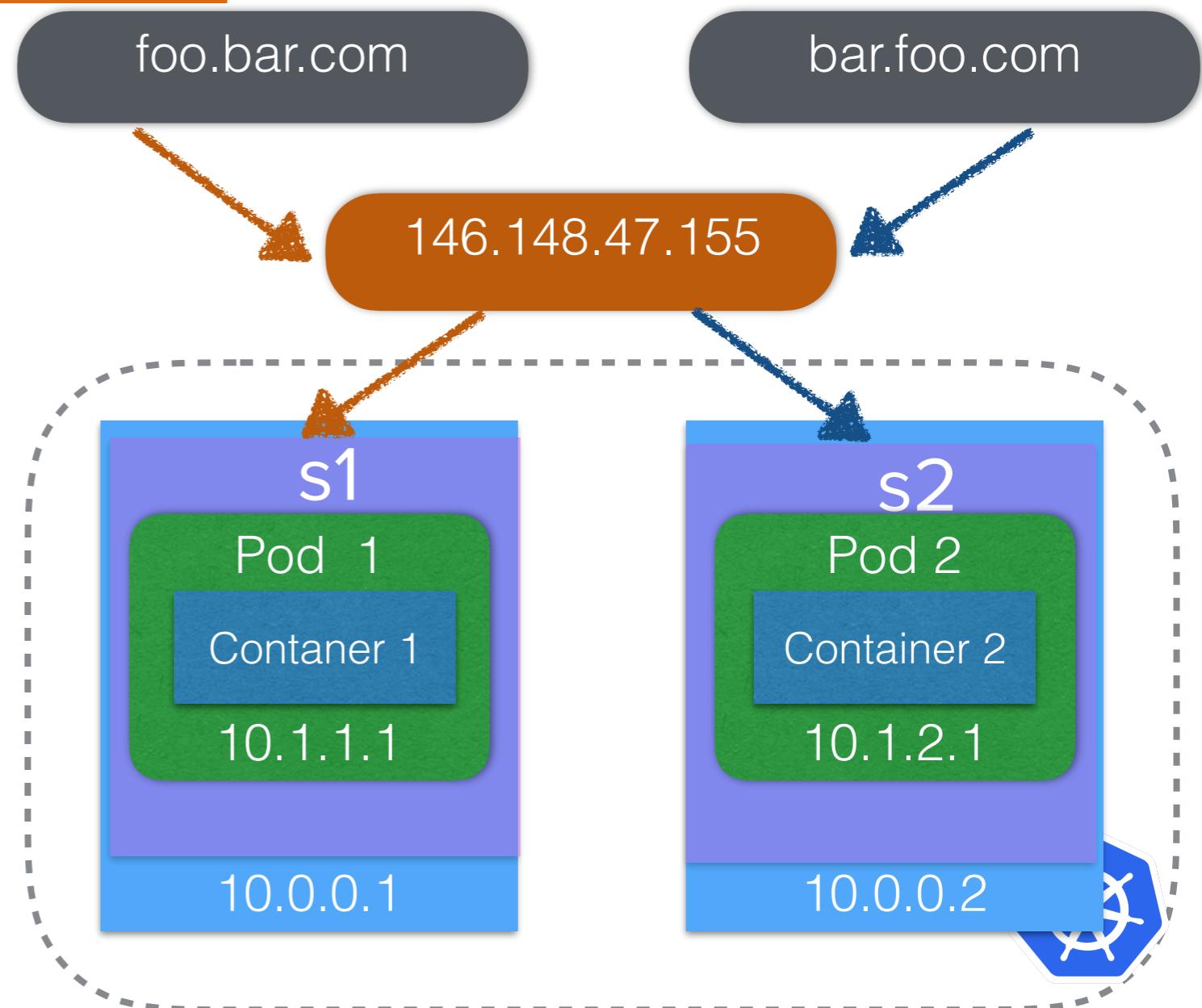
Simple fanout

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
  annotations:
    ingress.kubernetes.io/
rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: s1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: s2
          servicePort: 80
```



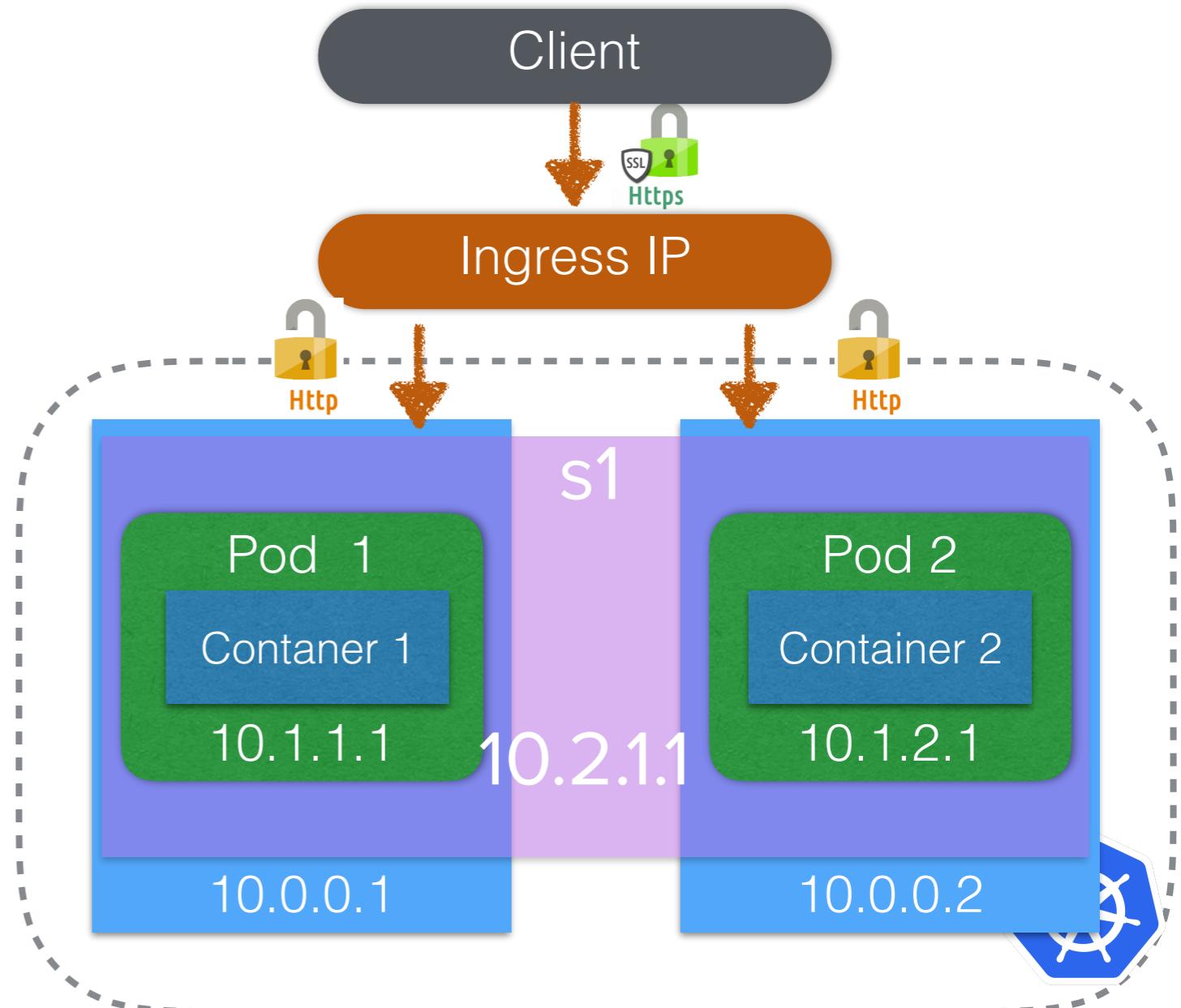
Name based virtual hosting

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: s1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: s2
          servicePort: 80
```



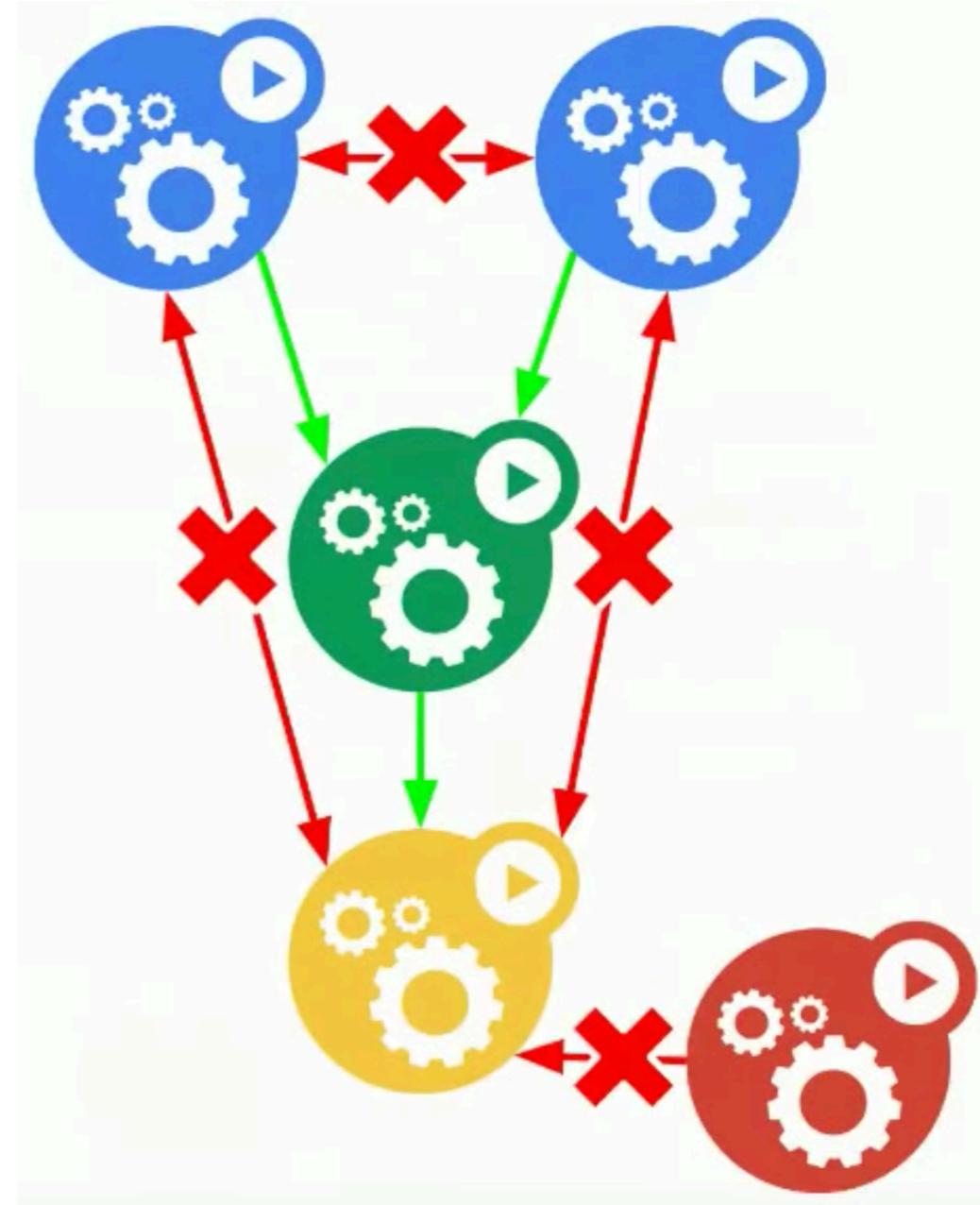
TLS termination

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: no-rules-map
spec:
  tls:
  - secretName: testsecret
  backend:
    serviceName: s1
    servicePort: 80
```

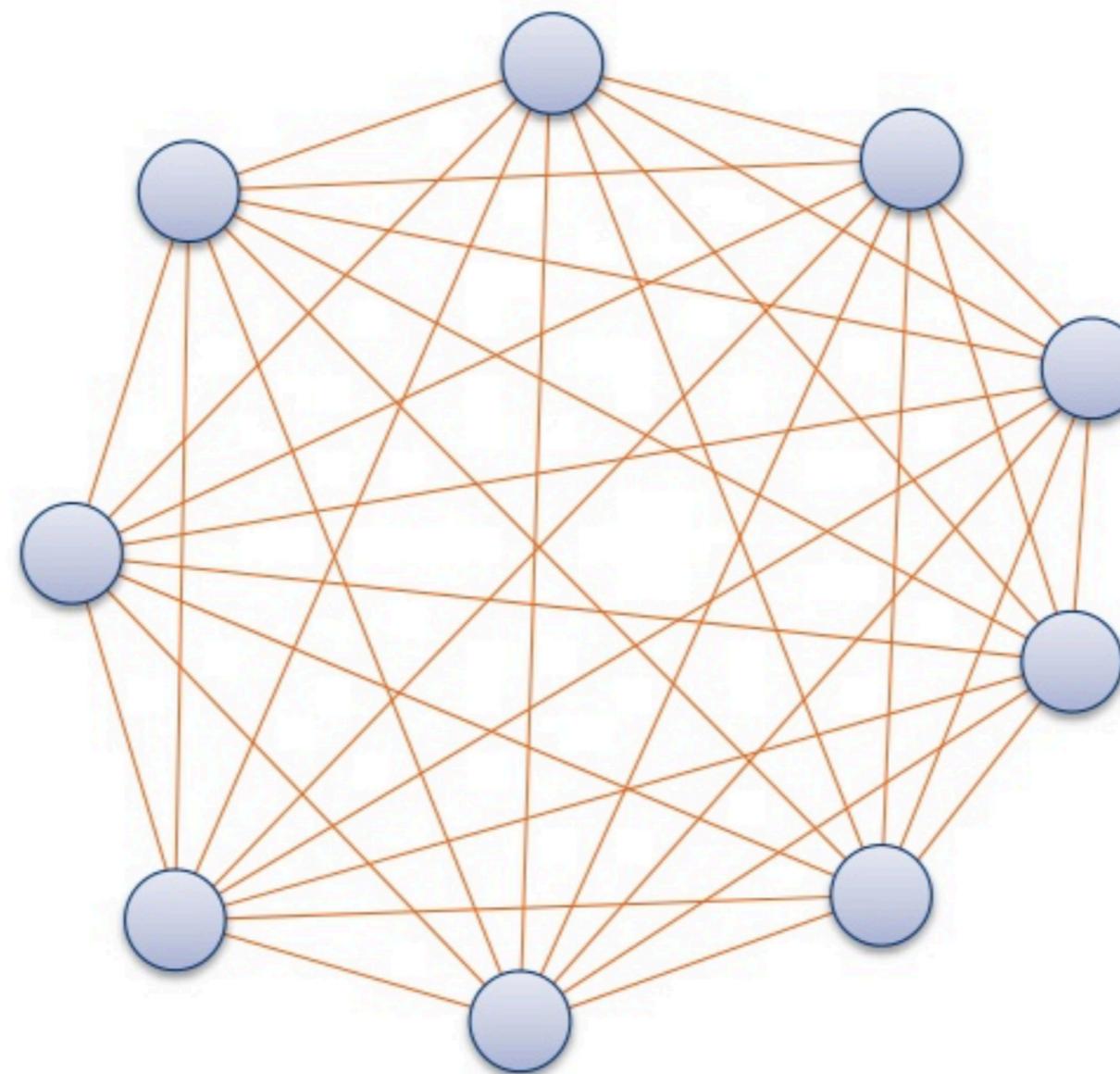


Network Policies

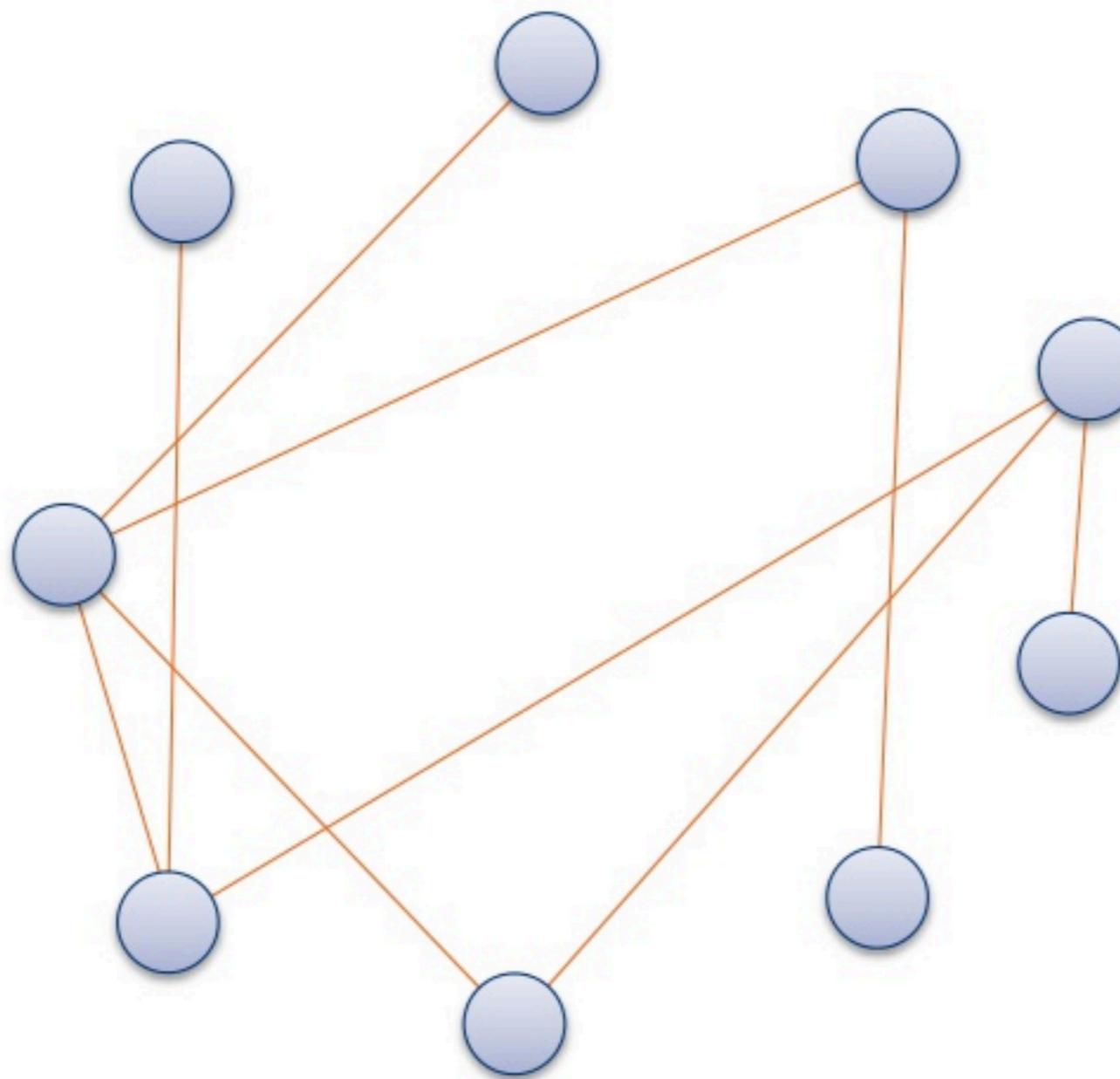
```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: access-nginx
spec:
  podSelector:
    matchLabels:
      run: nginx
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
```



Network isolation



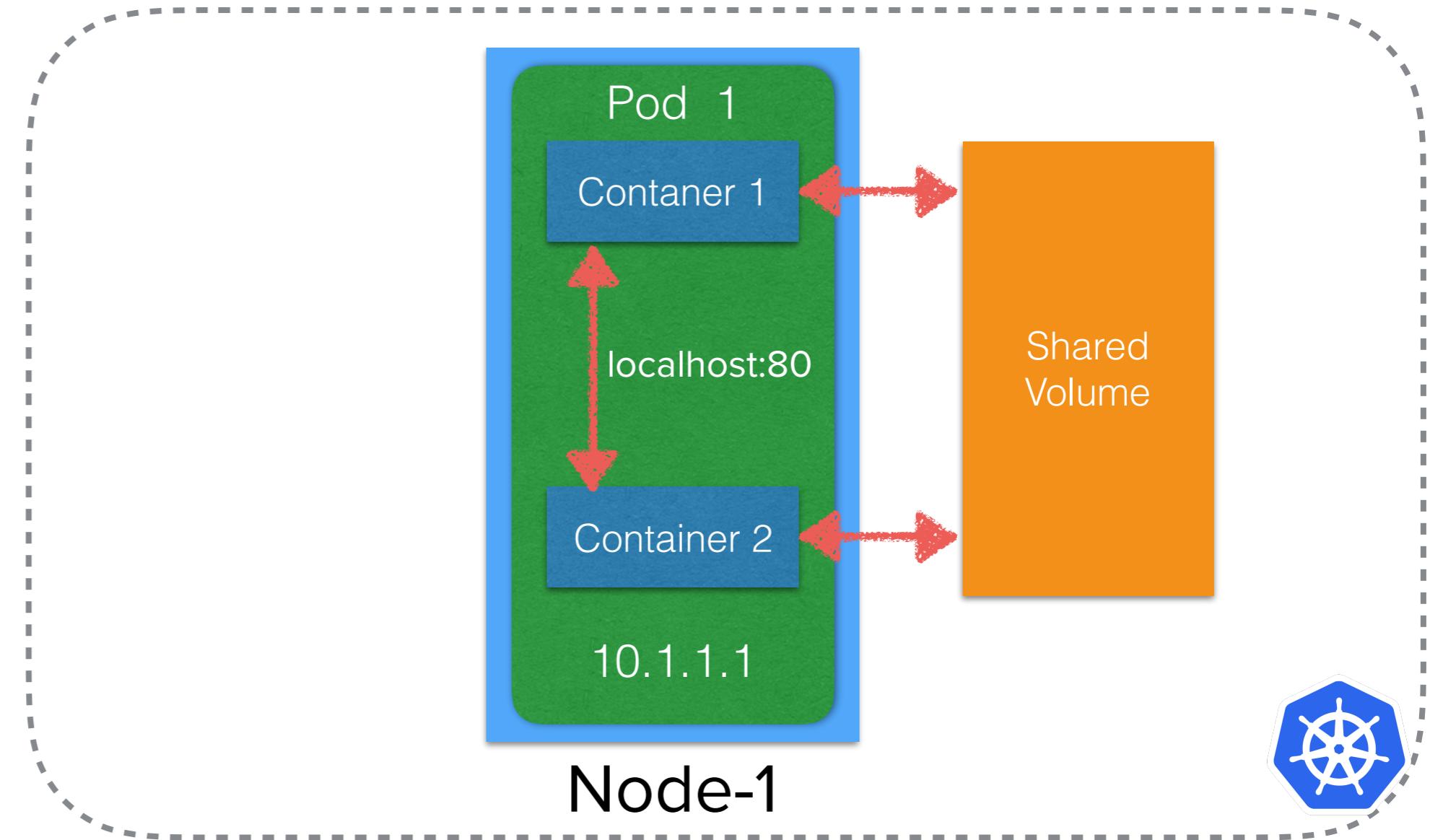
Network isolation



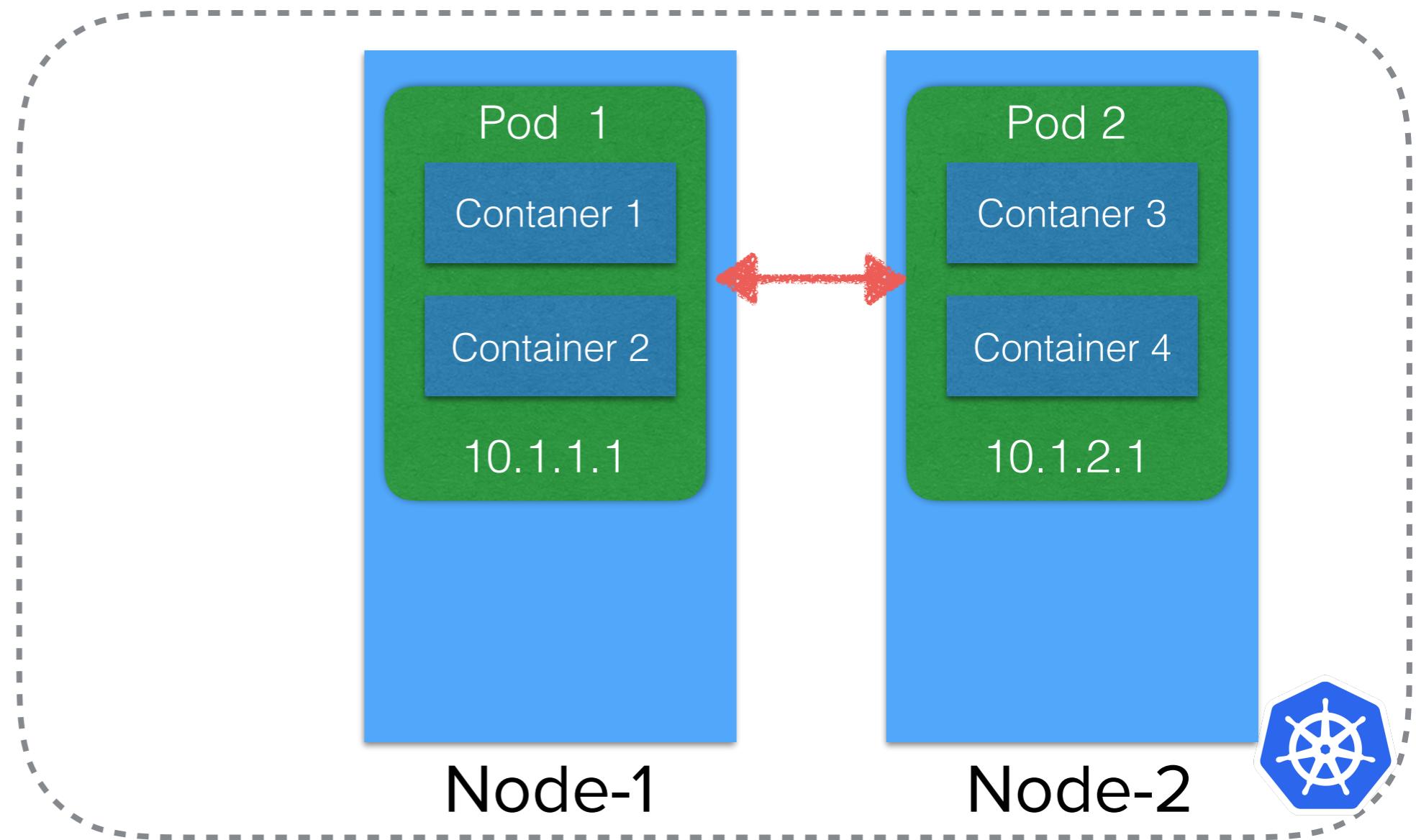
Kubernetes networking

- Методы взаимодействия
 - Контейнер-Контейнер
 - Pod-Pod
 - Pod-Service
 - Внешний трафик

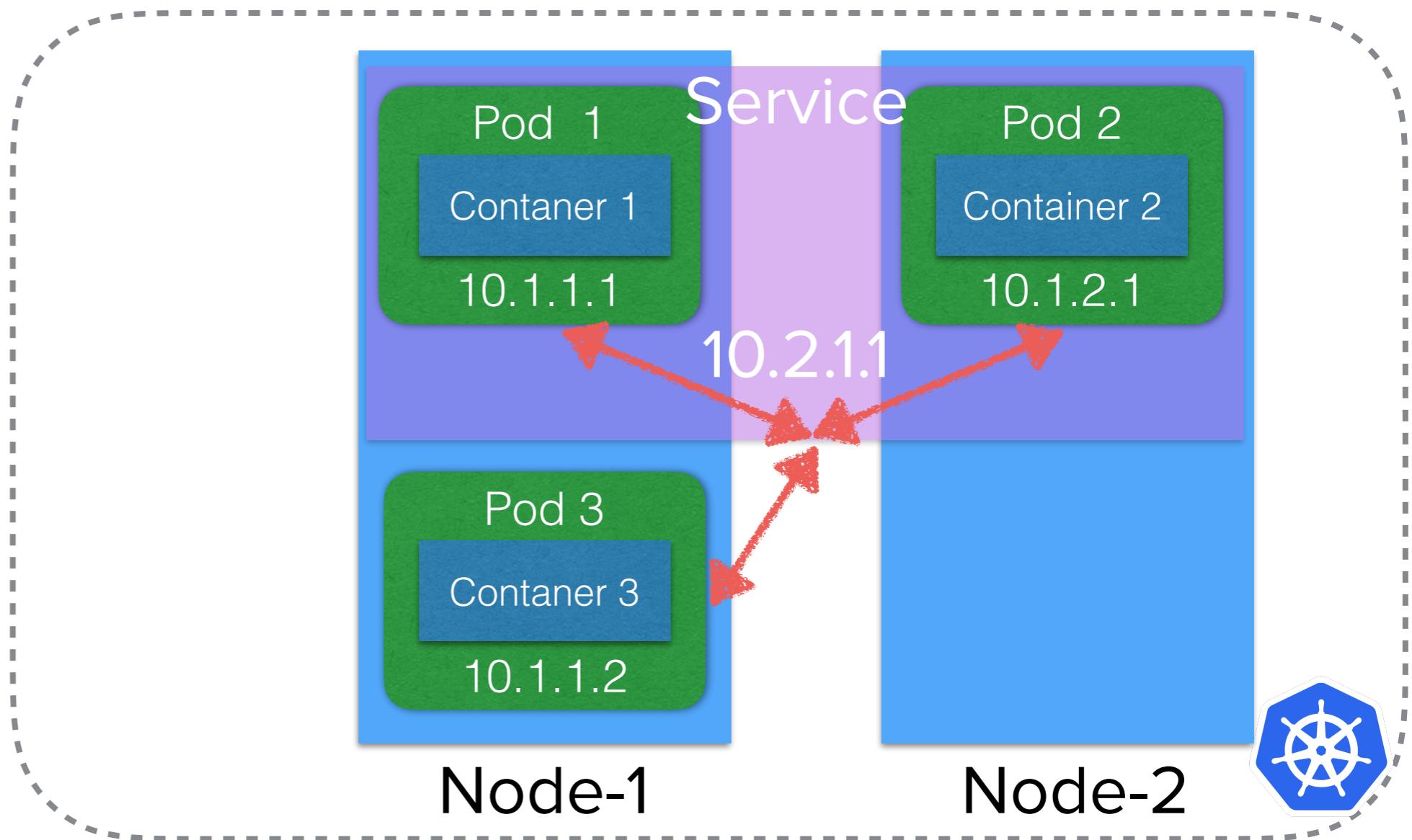
Контейнер - Контейнер



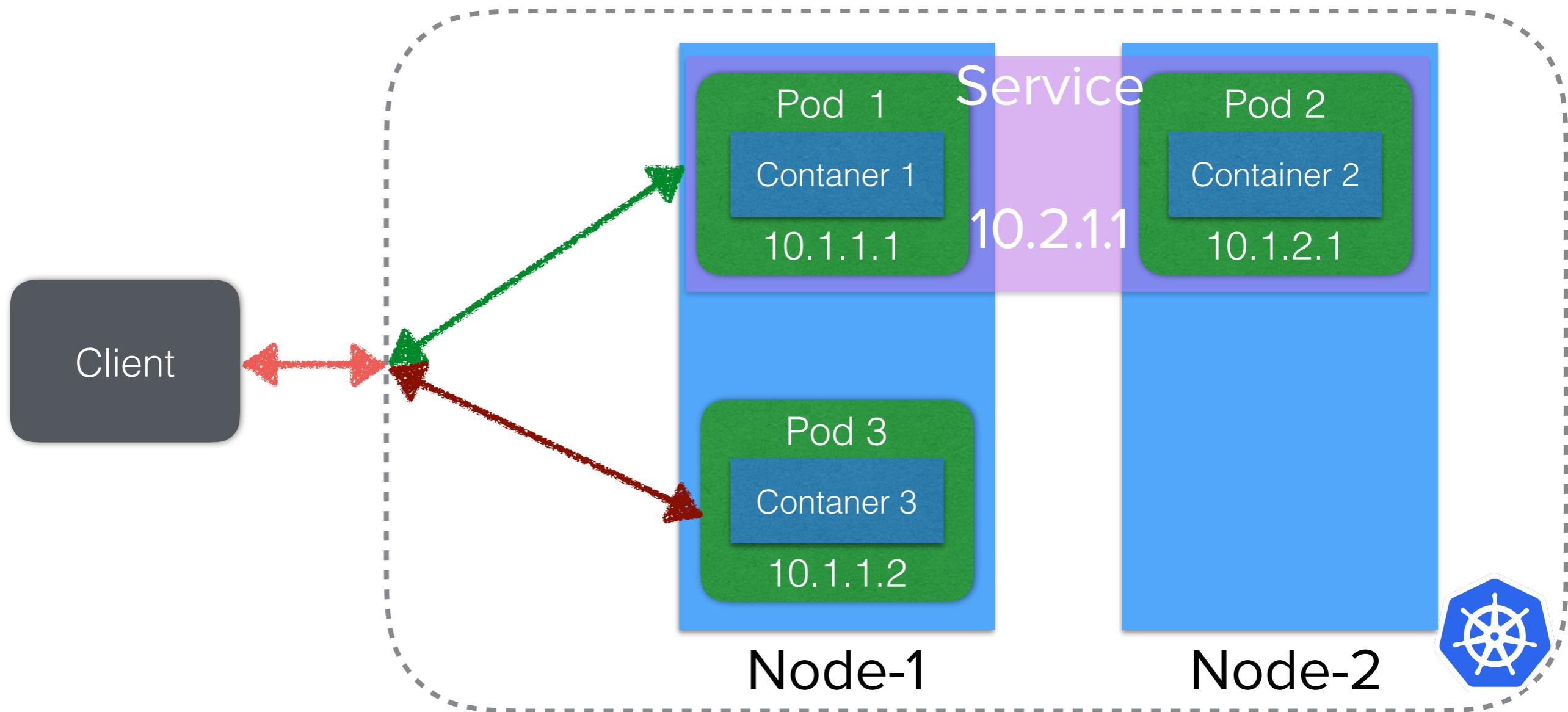
Pod - Pod



Pod - Service

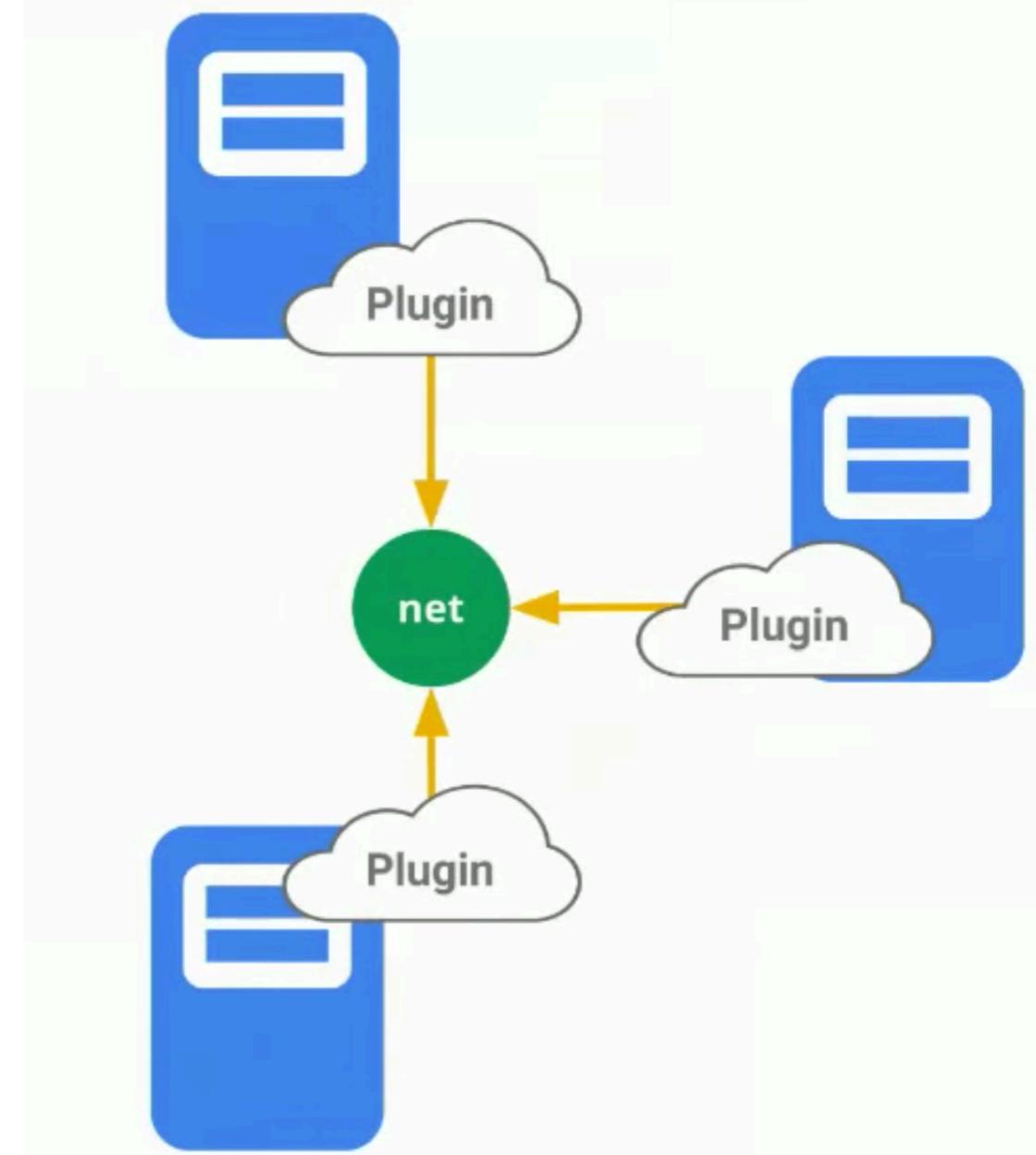


Внешний трафик



Network plugins

- kubenet
- AWS: Route tables
- Weave
- Calico
- Flannel
- OVS
- OpenContrail
- Cisco Contiv



Container Network Interface

- Спецификация для написания плагинов управляющих Linux Container'ами
- Поддерживается Cloud Native Computing Foundation

Хранение данных в Kubernetes

volumes

- Меньше ограничений по сравнению с Docker'ом
- Full lifecycle с привязкой к Pod'у
- Pod может использовать несколько типом Volume'ов одновременно

Local Volumes

- emptyDir - пустая директория, создается на Node'e и удаляется вместе с остановкой Pod'a
- hostPath - директория на Node'e
- local - локальное хранилище, диск, партиция или директория. Может быть создан только как PV

emptyDir

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
    - image: gcr.io/google_containers/test-webserver
      name: test-container
    volumeMounts:
      - mountPath: /cache
        name: cache-volume
  volumes:
    - name: cache-volume
      emptyDir: {}
```

Cloud Volumes

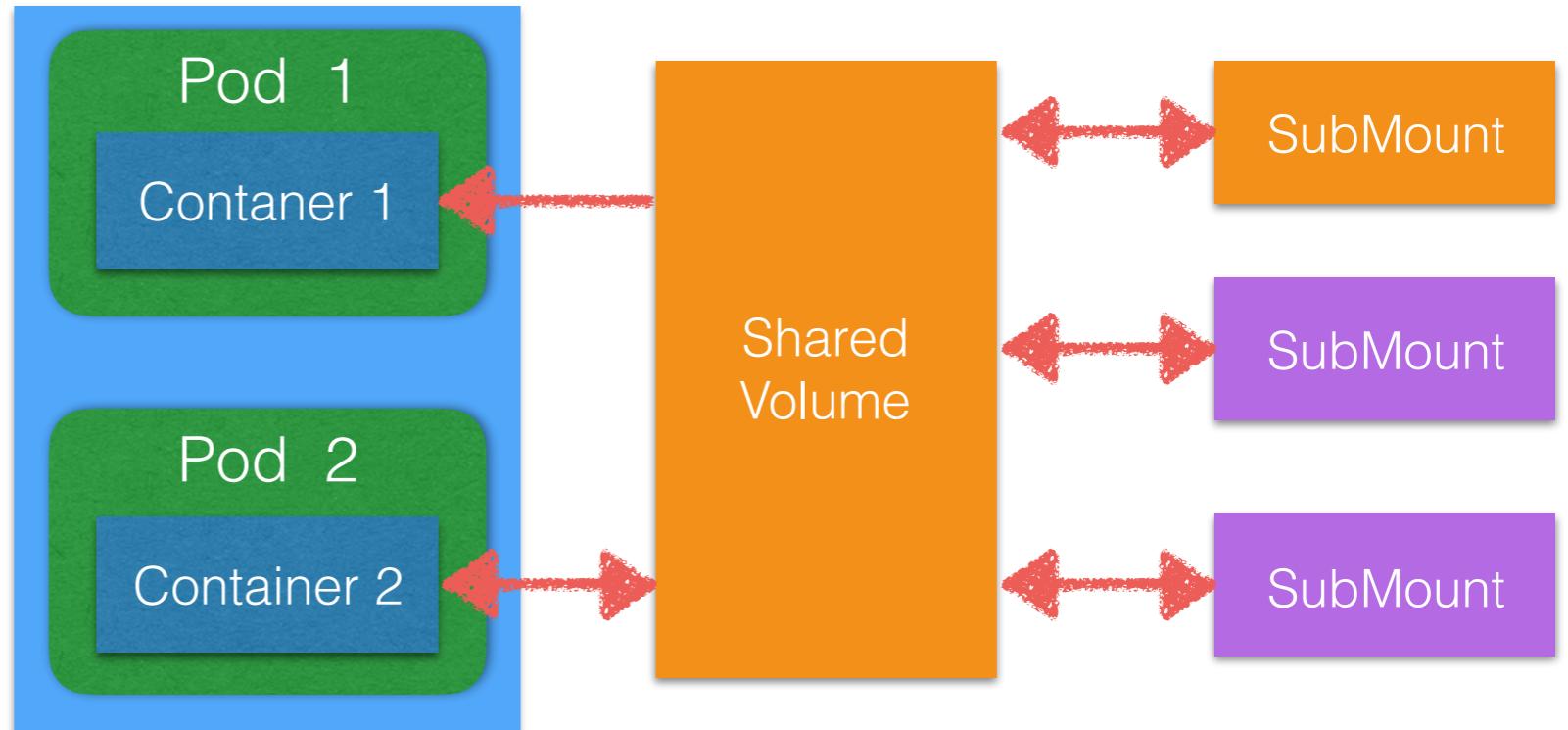
- AWS EBS
- Azure Disk and Azure File
- gcePersistentDisk

Custom Volumes

- cephfs and glusterfs
- fc and iscsi

Mount propagation

- HostToContainer
- Bidirectional



Persistent Volumes

- Позволяет отделить жизненный цикл контейнеров от данных
- PersistentVolume
- PersistentVolumeClaim

PersistentVolume

- Такой же ресурс кластера как и Node
- PV как обычные Volume'ы, но имеют отдельный от сервисов жизненный цикл
- Содержит всю информацию об имплементации хранилища

PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

PersistentVolume

- Capacity
- Access Modes - ReadWriteOnce/ReadOnlyMany/
ReadWriteMany
- Class
- Reclaim Policy - Retain/Recycle/Delete
- Mount Option
- Phase - Available/Bound/Released/Failed

PersistentVolumeClaim

- Запрос на хранилище от пользователя
- PVC могут требовать определенный объем хранилища и прав доступа

PersistentVolumeClaim

- Access Modes
- Resources
- Selector
- Class

PersistentVolumeClaim

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
  matchExpressions:
    - {key: environment, operator: In, values: [dev]}
```

PVC as Volume

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
        - mountPath: "/var/www/html"
          name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

PV portability

- В конфигурации описывайте PVC вместо PV
- Предпочитительно использование Storage Class
- Следите за не выполненными PVC

Storage Classes

- Описание "классов" различных систем хранения
- Разные классы могут использоваться для:
 - QoS уровни
 - Произвольных политик
 - Для динамического provisioning

Storage Classes

- Provisioner
- Reclaim Policy
- Mount Options

Storage Classes

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
mountOptions:
  - debug
```

Dynamic Volume Provisioning

- Использует StorageClass для provisioning'a
- PersistentVolumeClaim содержит информацию о нужном StorageClass

Lifecycle of a volume and claim

- Provisioning:
 - Static
 - Dynamic
- Binding
- Using
- Reclaiming