

Устройство кластера, сеть, отказоустойчивость

План

- Компоненты кластера k8s и их взаимодействие
- Сети в Kubernetes
- Отказоустойчивость кластера



Компоненты кластера

- Etcd
- API server
- Controller-manager
- Scheduler
- Kubelet
- Kube-proxy

Компоненты кластера

- Etcd
- API server
- Controller-manager
- Scheduler
- Kubelet
- Kube-proxy
- Контейнеризация
- Сеть
- DNS

Компоненты кластера

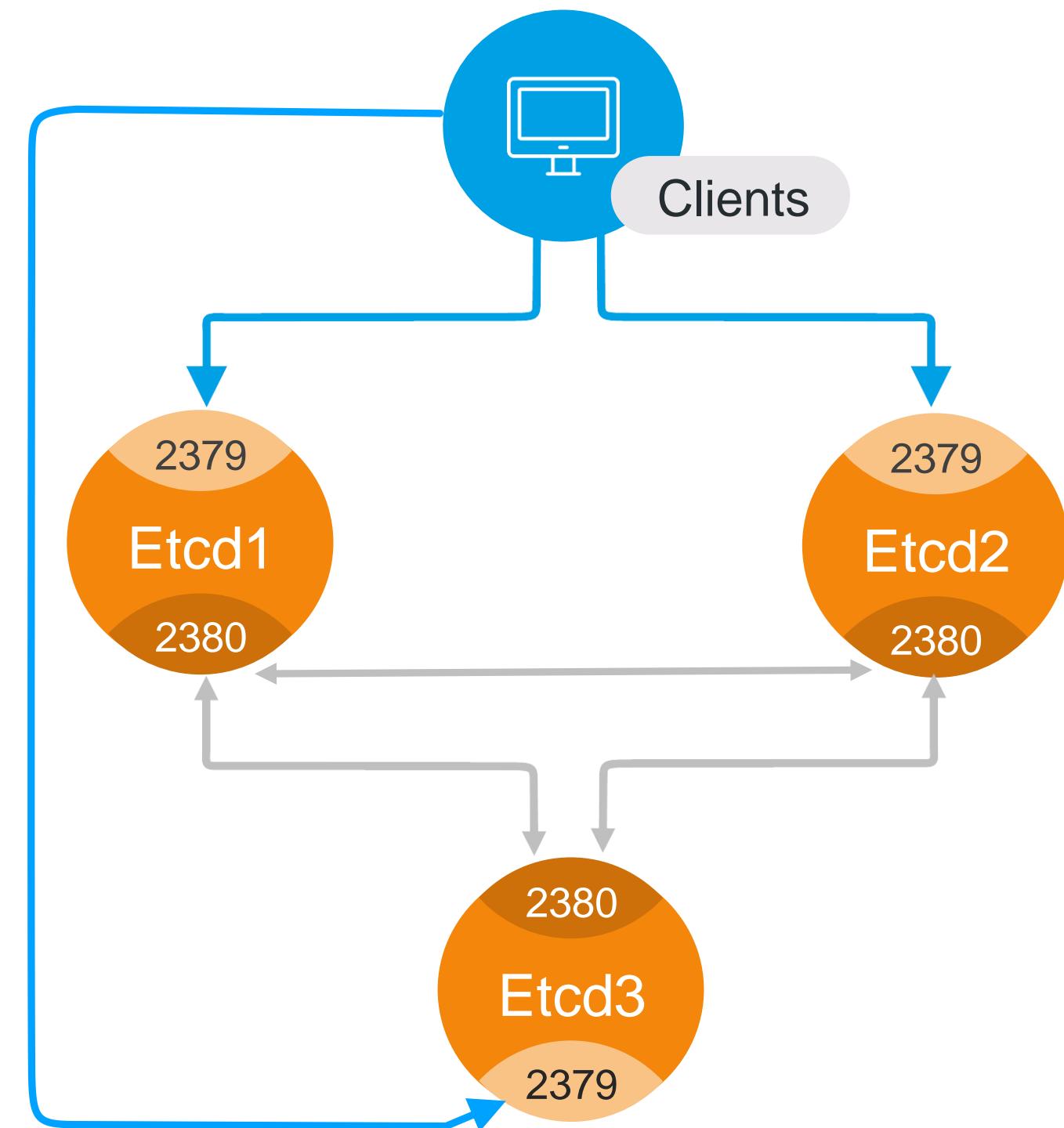
- Etcd
- API server
- Controller-manager
- Scheduler
- Kubelet
- Kube-proxy



Кто отдает команды
для запуска приложения?

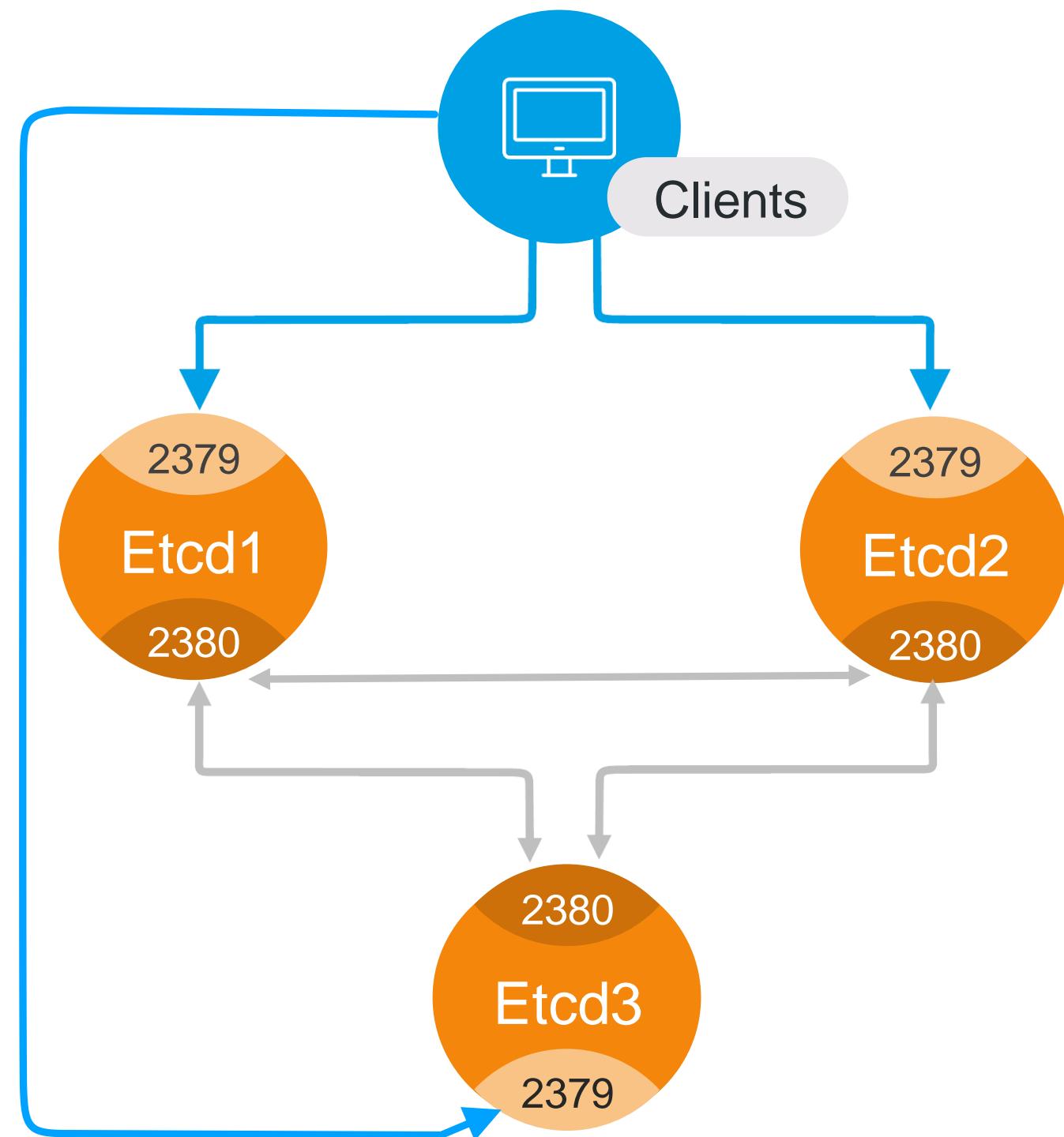
Etcd

- Хранит всю информацию о кластере



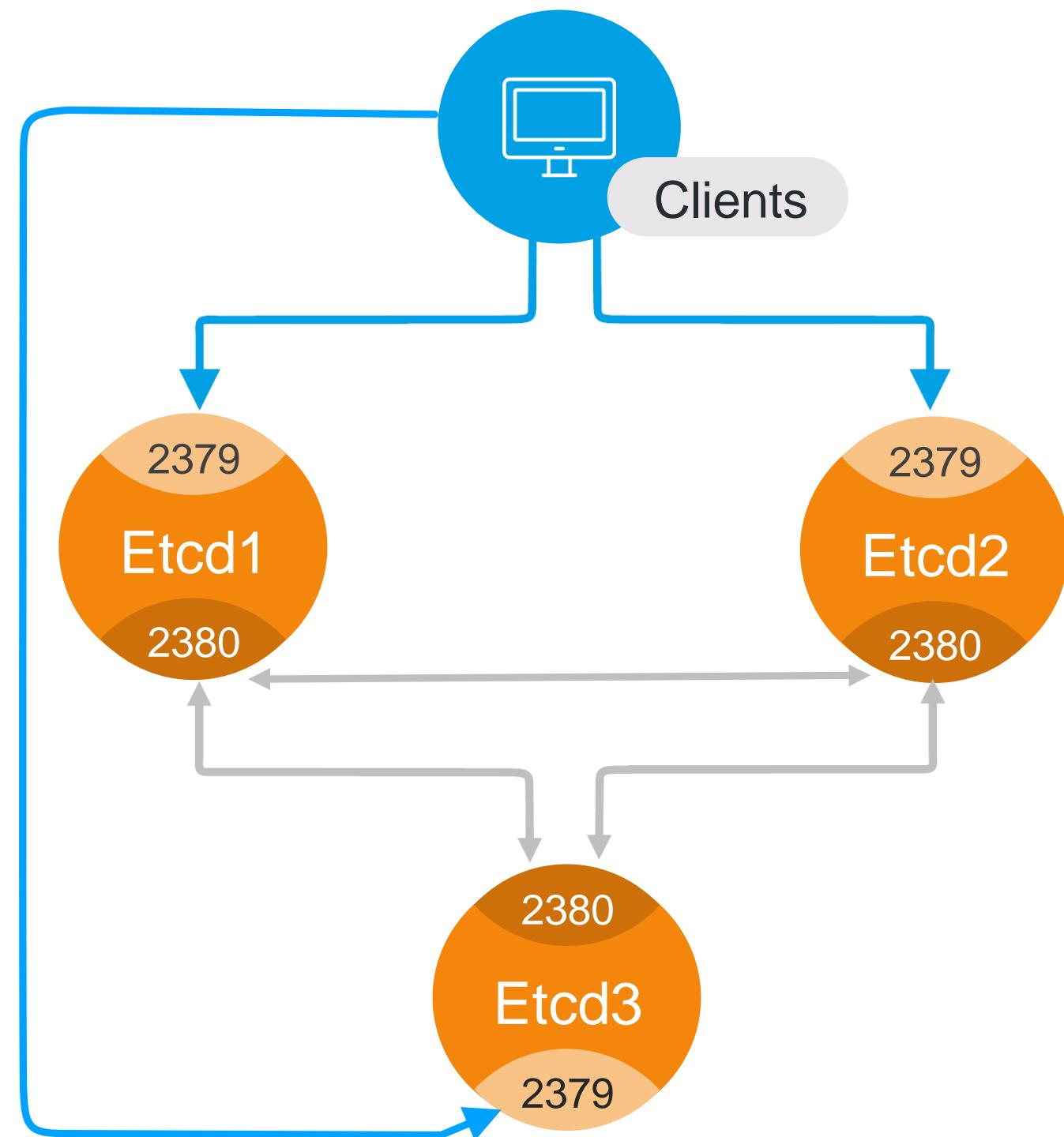
Etcd

- Хранит всю информацию о кластере
- Etcdctl – утилита управления кластером ETCD



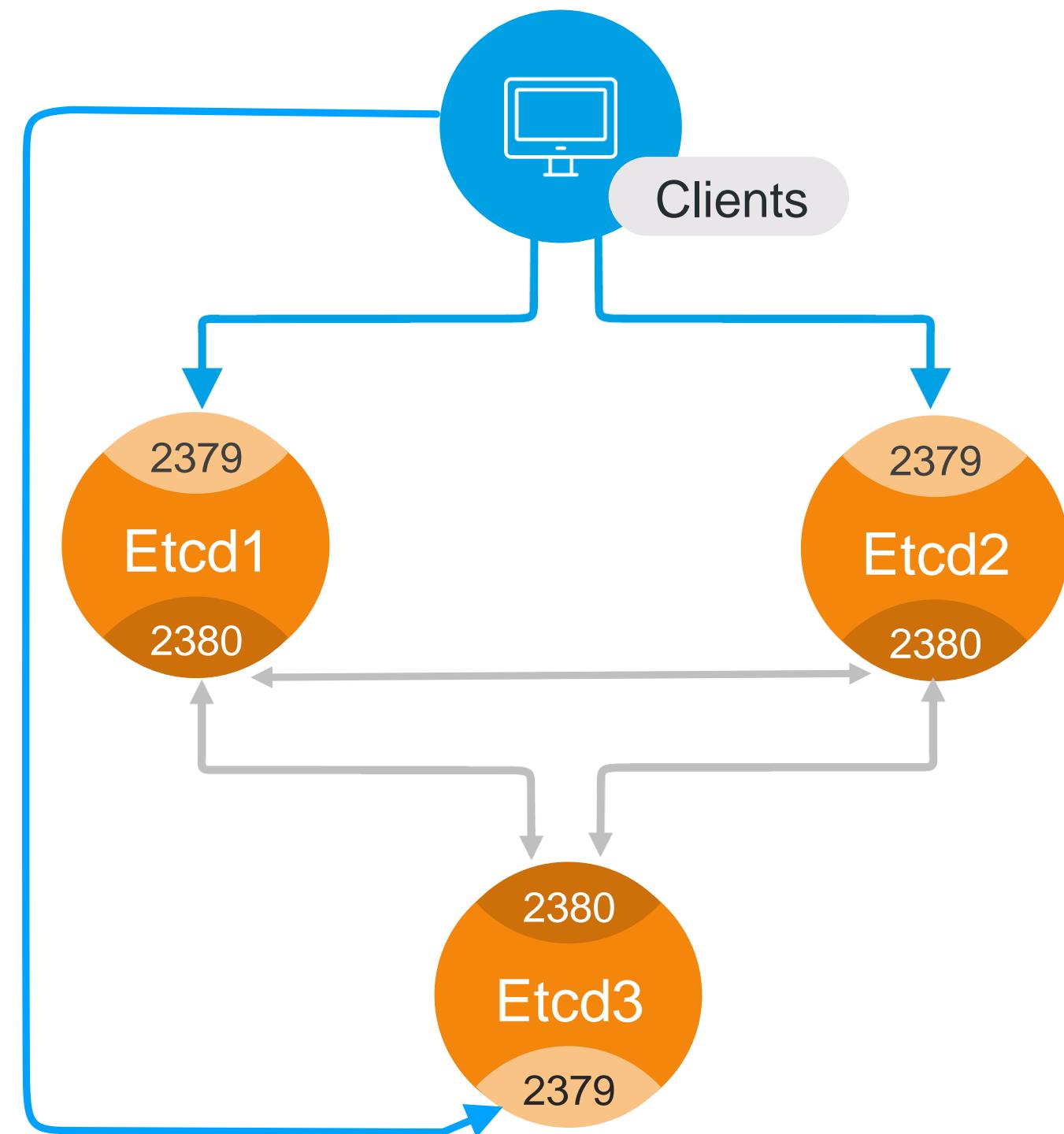
Etcd

- Хранит всю информацию о кластере
- Etcdctl – утилита управления кластером ETCD
- У ETCD две версии API v2/v3



Etcd

- Хранит всю информацию о кластере
- Etcdctl – утилита управления кластером ETCD
- У ETCD две версии API v2/v3
- Требует быстрых дисков



API server

- Центральный компонент Kubernetes



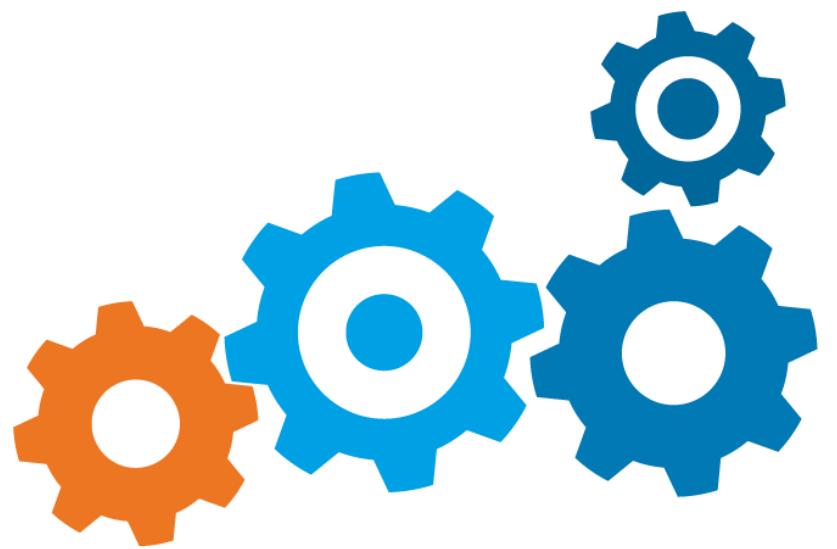
API server

- Центральный компонент Kubernetes
- Единственный кто общается с Etcd



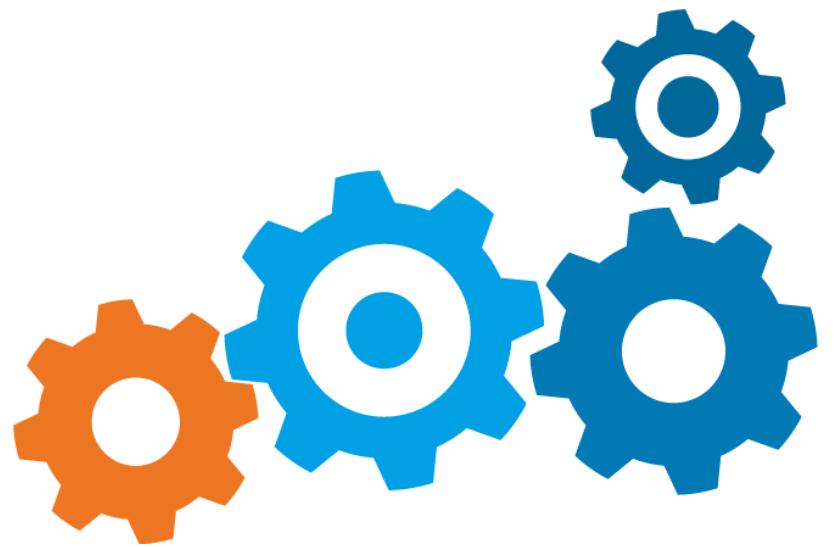
API server

- Центральный компонент Kubernetes
- Единственный кто общается с Etcd
- Работает по REST API

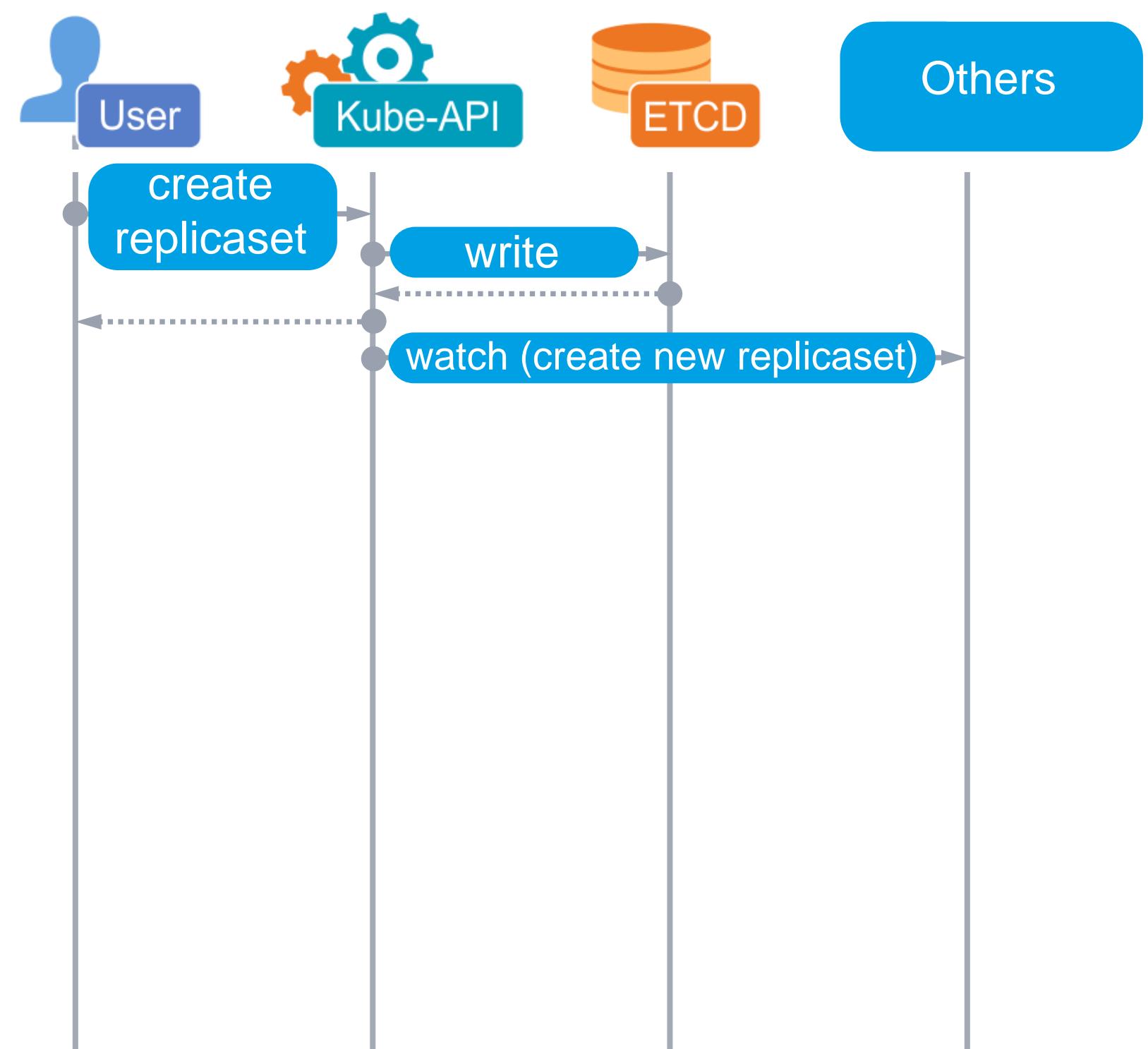


API server

- Центральный компонент Kubernetes
- Единственный кто общается с Etcd
- Работает по REST API
- Authentication and authorization



API server



Controller-manager

- Набор контроллеров



Controller-manager

- Набор контроллеров
 - Node controller



Controller-manager

- Набор контроллеров
 - Node controller
 - Replication controller



Controller-manager

- Набор контроллеров
 - Node controller
 - Replication controller
 - Endpoints controller



Controller-manager

- Набор контроллеров
 - Node controller
 - Replication controller
 - Endpoints controller
 - И другие...

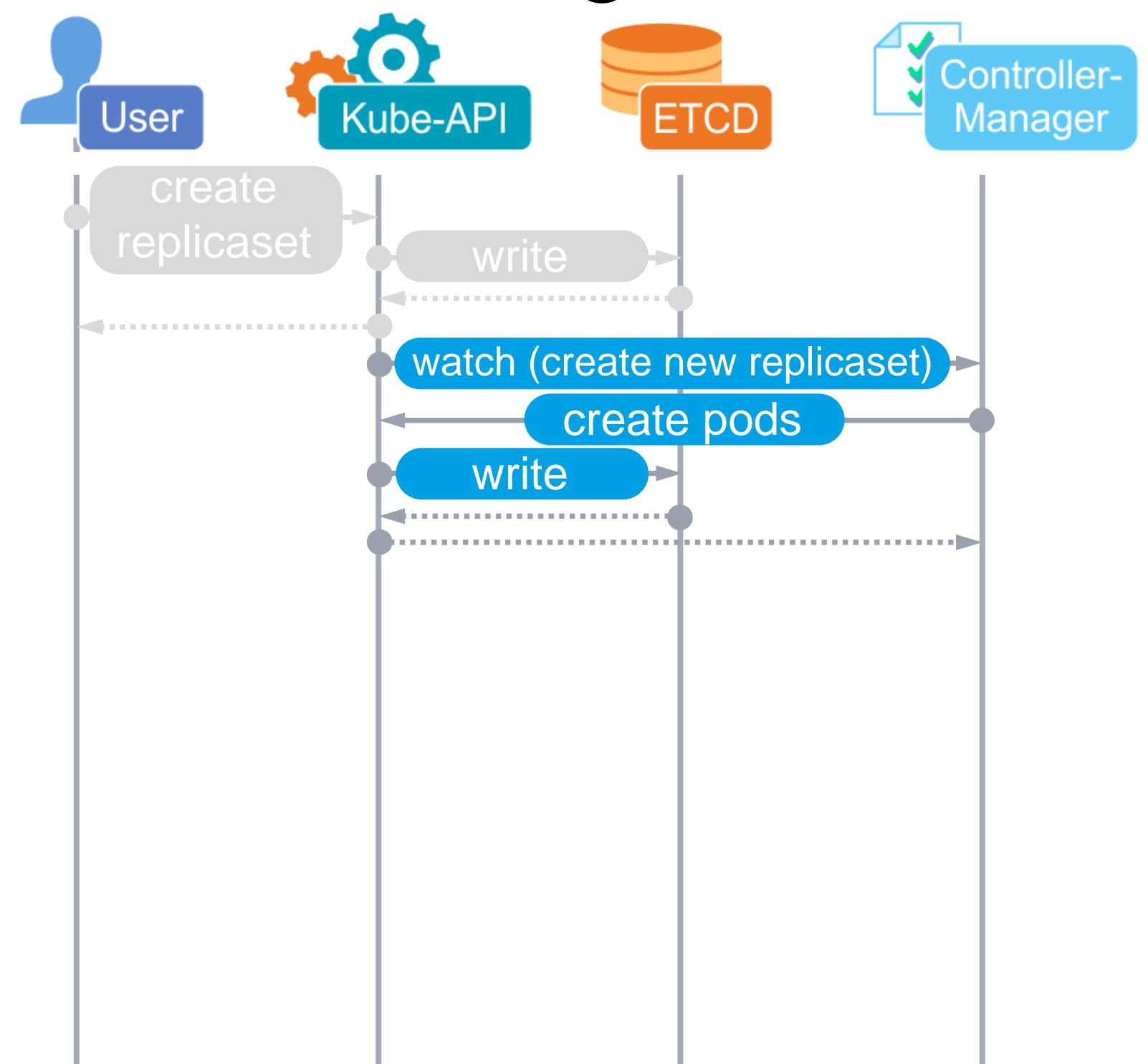


Controller-manager

- Набор контроллеров
 - Node controller
 - Replication controller
 - Endpoints controller
 - И другие...
- GarbageCollector – «сборщик мусора»

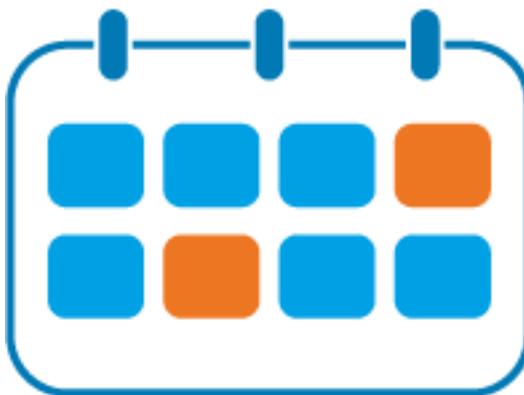


Controller-manager



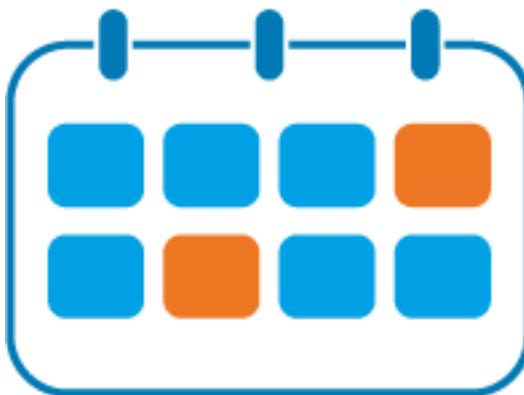
Scheduler

- Назначает РОДы на ноды, учитывая:



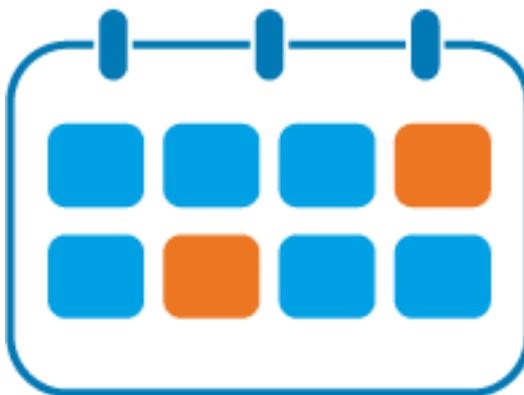
Scheduler

- Назначает PODы на ноды, учитывая:
 - QoS



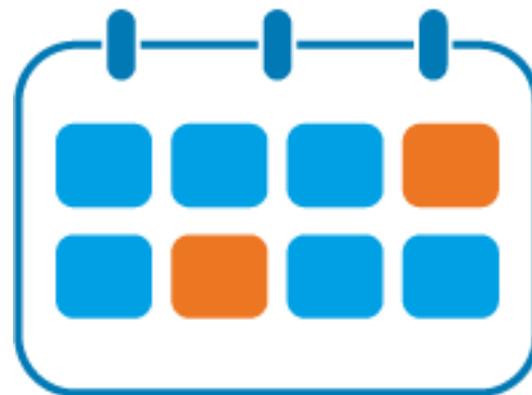
Scheduler

- Назначает РОДы на ноды, учитывая:
 - QoS
 - Affinity / anti-affinity

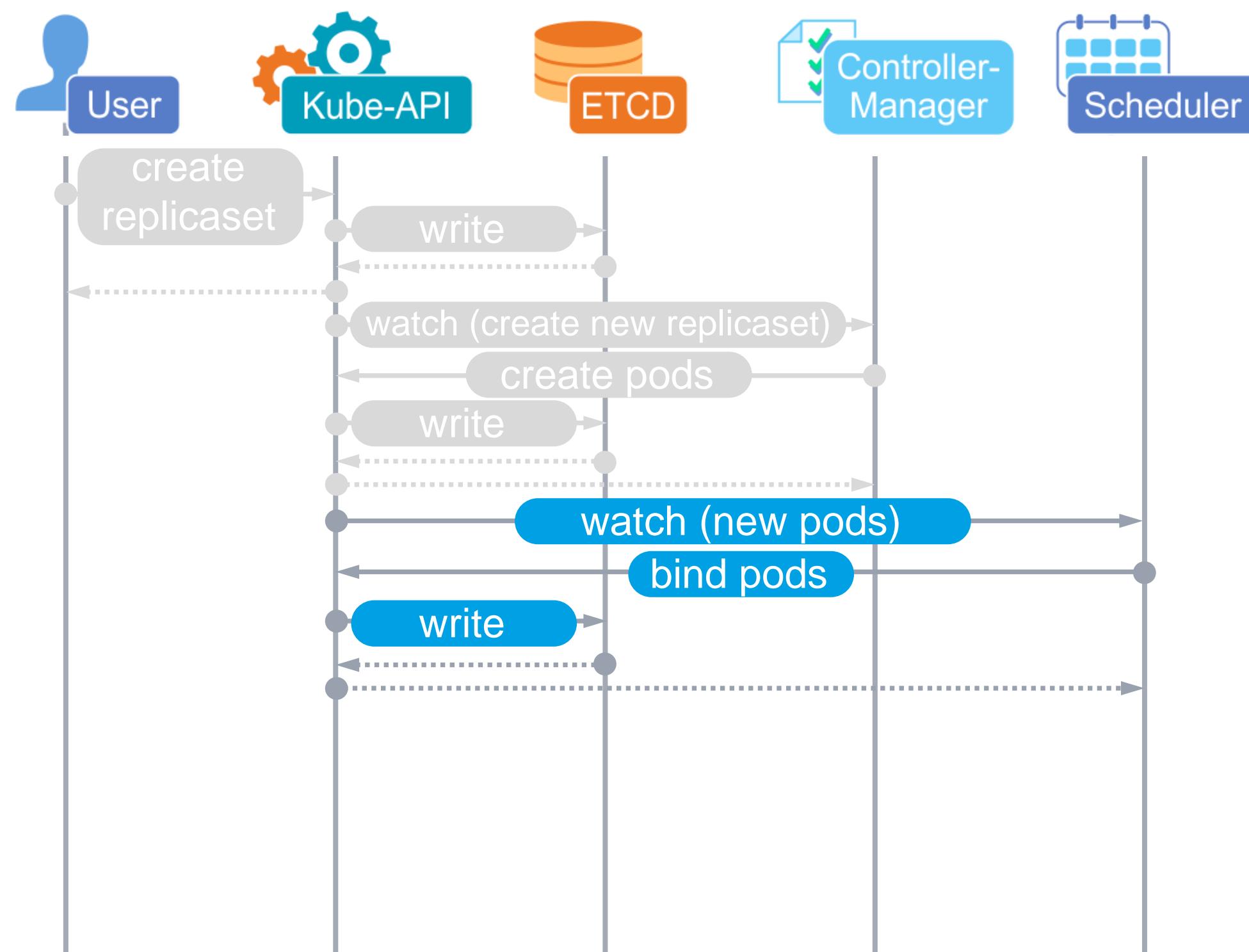


Scheduler

- Назначает РОДы на ноды, учитывая:
 - QoS
 - Affinity / anti-affinity
 - Requested resources



Scheduler



Kubelet

- Работает на каждой ноде



Kubelet

- Работает на каждой ноде
- Единственный компонент, работающий не в Docker



Kubelet

- Работает на каждой ноде
- Единственный компонент, работающий не в Docker
- Отдает команды Docker daemon

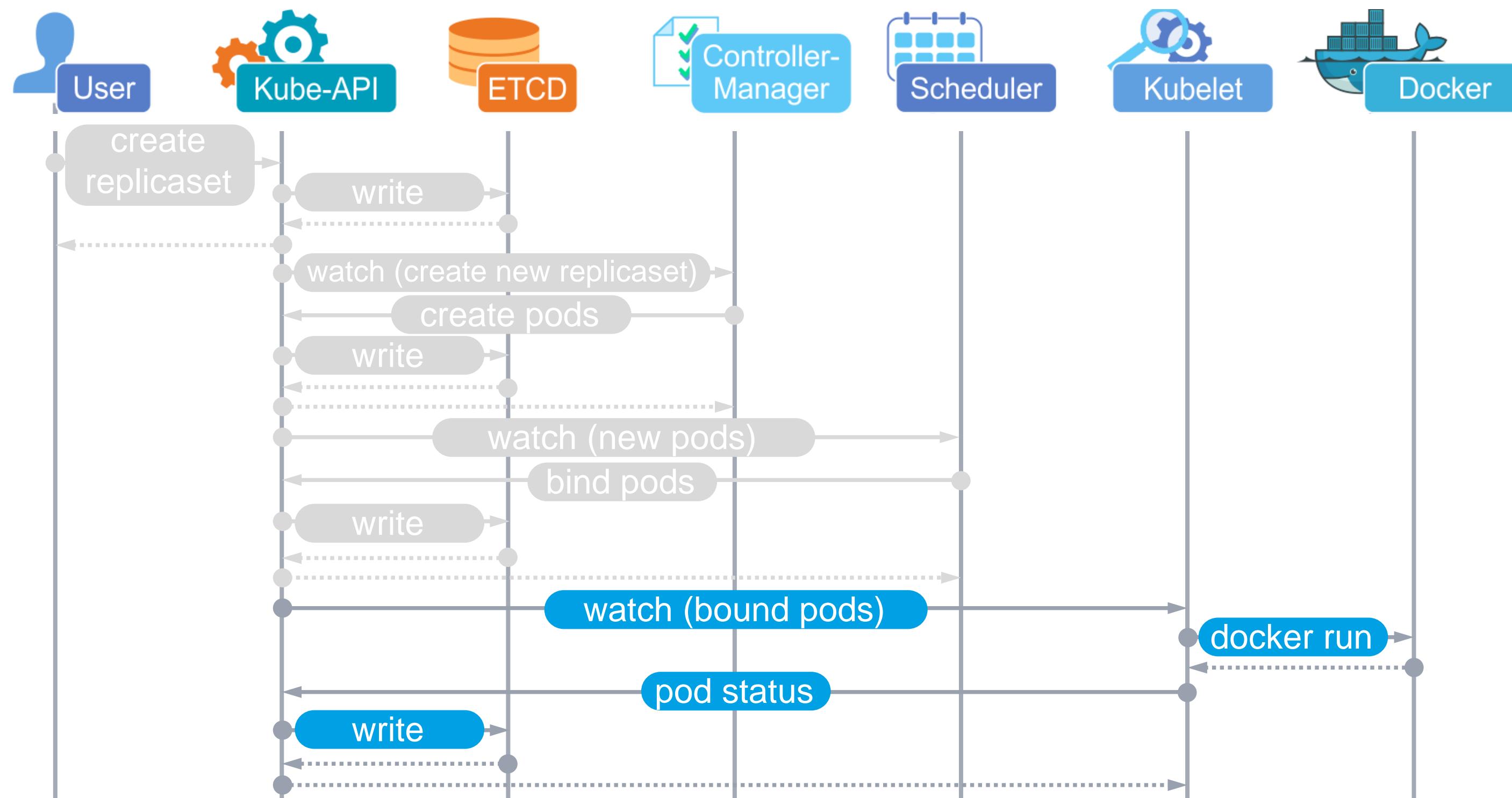


Kubelet

- Работает на каждой ноде
- Единственный компонент, работающий не в Docker
- Отдает команды Docker daemon
- Создает PODы



Kubelet

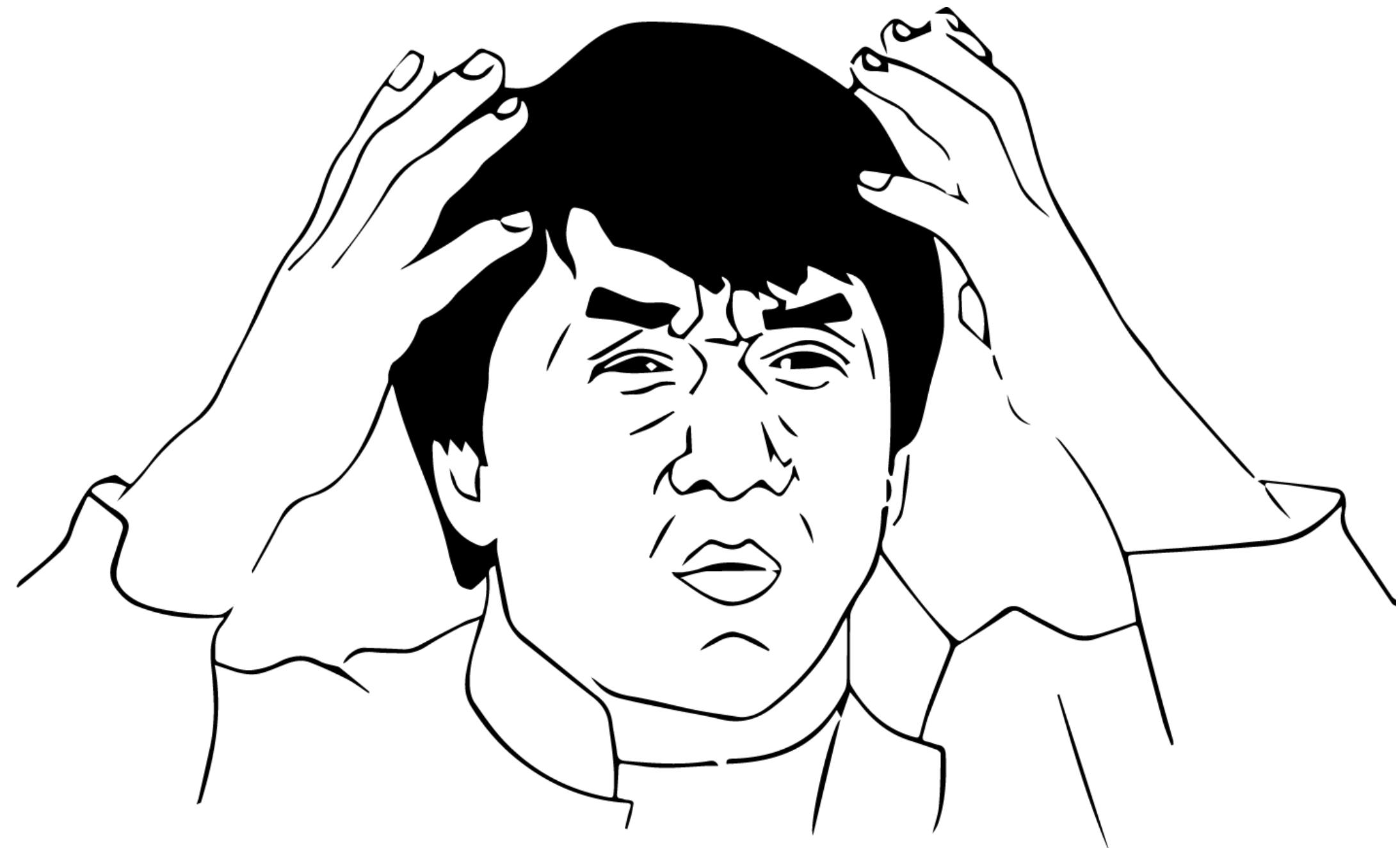


Компоненты кластера

- Etcd
- API server
- Controller-manager
- Scheduler
- Kubelet
- Kube-proxy

Кто **отдает команды**
для запуска приложения?

Зачем мне это всё?



Компоненты кластера

- Etcd
- API server
- Controller-manager
- Scheduler
- Kubelet
- Kube-proxy



Kube-proxy

- Смотрит в Kube-API

Kube-proxy

- Сматривает в Kube-API
- Стоит на всех серверах

Kube-proxy

- Сматривает в Kube-API
- Стоит на всех серверах
- Управляет сетевыми правилами на нодах

Kube-proxy

- Сматривает в Kube-API
- Стоит на всех серверах
- Управляет сетевыми правилами на нодах
- Фактически реализует Service (ipvs и iptables)

Service

```
-A KUBE-SERVICES
  -d 1.1.1.1/32
  -p tcp
  -m comment --comment "mynamespace/myservice:http cluster
IP"
  -m tcp --dport 80
-j KUBE-SVC-UT6A43GJFBEDB03V
```

Service

```
-A KUBE-SERVICES
  -d 1.1.1.1/32
  -p tcp
  -m comment --comment "mynamespace/myservice:http cluster
IP"
  -m tcp --dport 80
-j KUBE-SVC-UT6A43GJFBEDB03V

-A KUBE-SVC-UT6A43GJFBEDB03V
  -m comment --comment "mynamespace/myservice:http"
  -m statistic
    --mode random --probability 0.5000000000
-j KUBE-SEP-MMYWB6DZJI4RZ5CQ

-A KUBE-SVC-UT6A43GJFBEDB03V
  -m comment --comment "mynamespace/myservice:http"
-j KUBE-SEP-J33LX377GA3DLDWM
```

Service

```
-A KUBE-SVC-UT6A43GJFBEDB03V
  -m comment --comment "mynamespace/myservice:http"
-j KUBE-SEP-J33LX377GA3DLDWM

-A KUBE-SEP-J33LX377GA3DLDWM
-p tcp
-m comment --comment "mynamespace/myservice:http"
-m tcp
-j DNAT
--to-destination 10.102.3.49:80
```

Service

```
-A KUBE-SVC-UT6A43GJFBEDB03V
  -m comment --comment "mynamespace/myservice:http"
  -m statistic
    --mode random --probability 0.5000000000
-j KUBE-SEP-J33LX377GA3DLDWM

-A KUBE-SEP-J33LX377GA3DLDWM
-p tcp
-m comment --comment "mynamespace/myservice:http"
-m tcp
-j DNAT
--to-destination 10.102.3.49:80
```

Service

```
$ kubectl get po --namespace=mynamespace -o wide
```

pod-1	1/1	Running	0	6h	10.102.3.49
pod-2	1/1	Running	0	6h	10.102.0.93

Почему не пингуется Service?

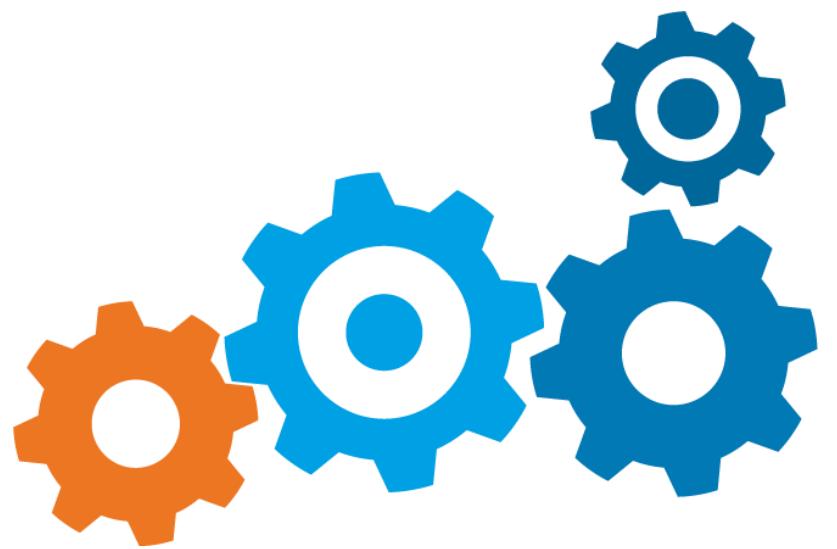
Service

- Статический IP



Service

- Статический IP
- DNS имя в kube-dns на этот IP
(myservice.mynamespace.svc.cluster.local)



Service

- Статический IP
- DNS имя в kube-dns на этот IP
(myservice.mynamespace.svc.cluster.local)
- Правила iptables для роутинга



Service

- Статический IP
- DNS имя в kube-dns на этот IP
(myservice.mynamespace.svc.cluster.local)
- Правила iptables для роутинга
- Service это не прокси!

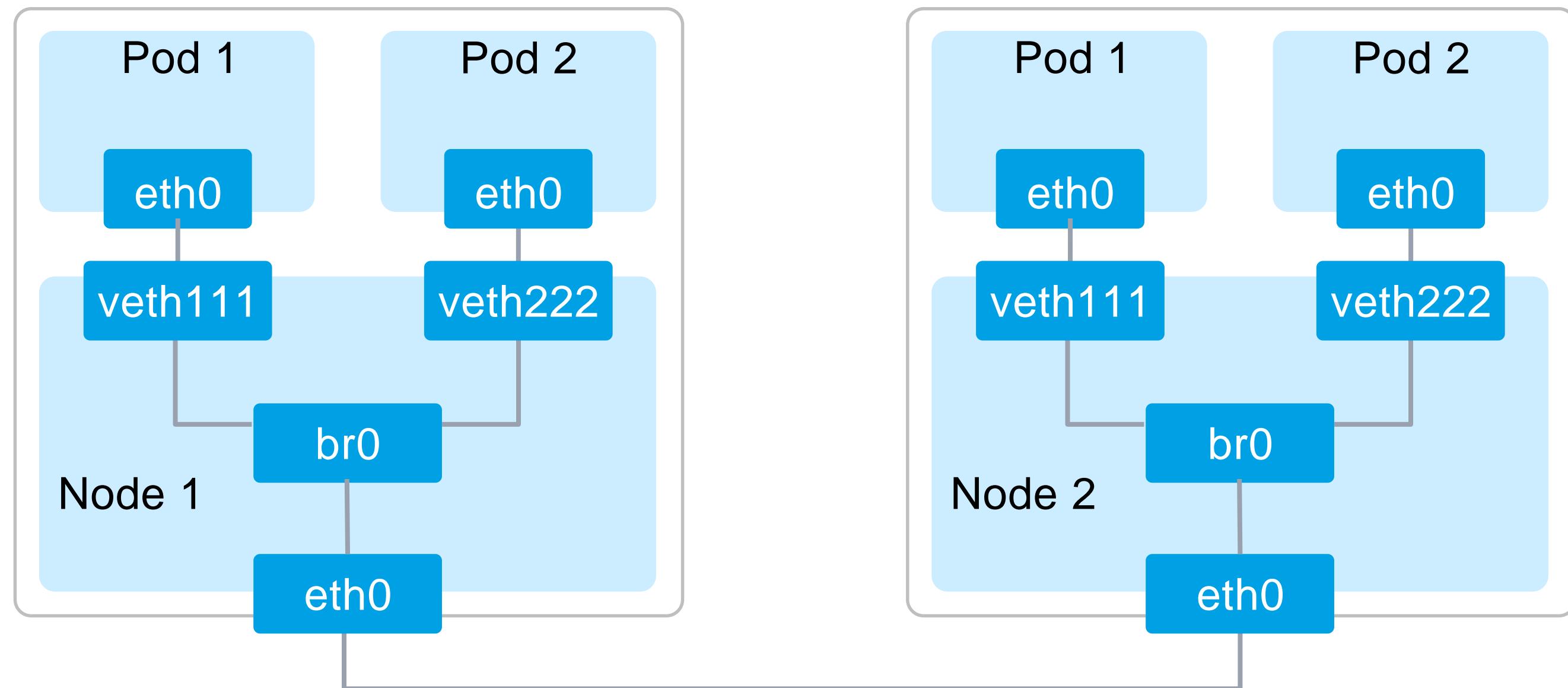


Service

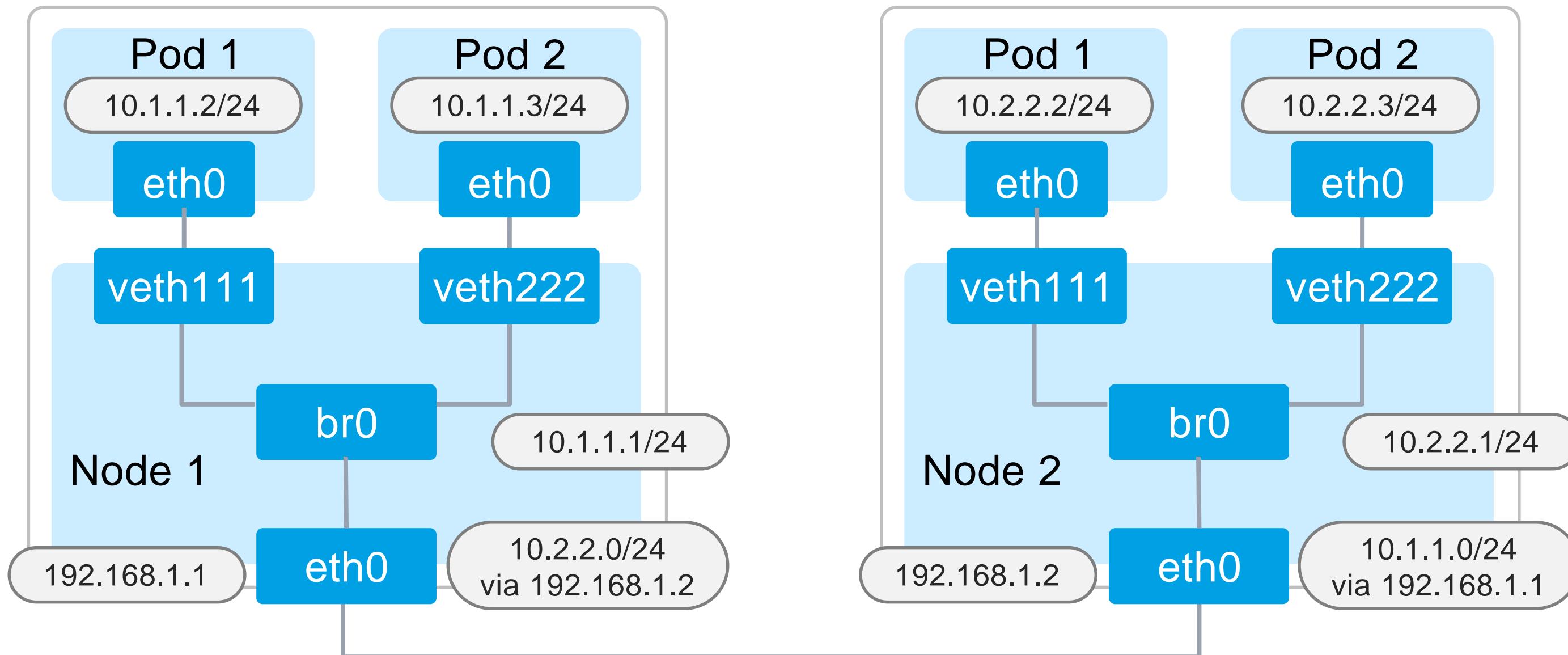
- Статический IP
- DNS имя в kube-dns на этот IP
(myservice.mynamespace.svc.cluster.local)
- Правила iptables для роутинга
- Service это не прокси!
- Проблемы NAT в Linux



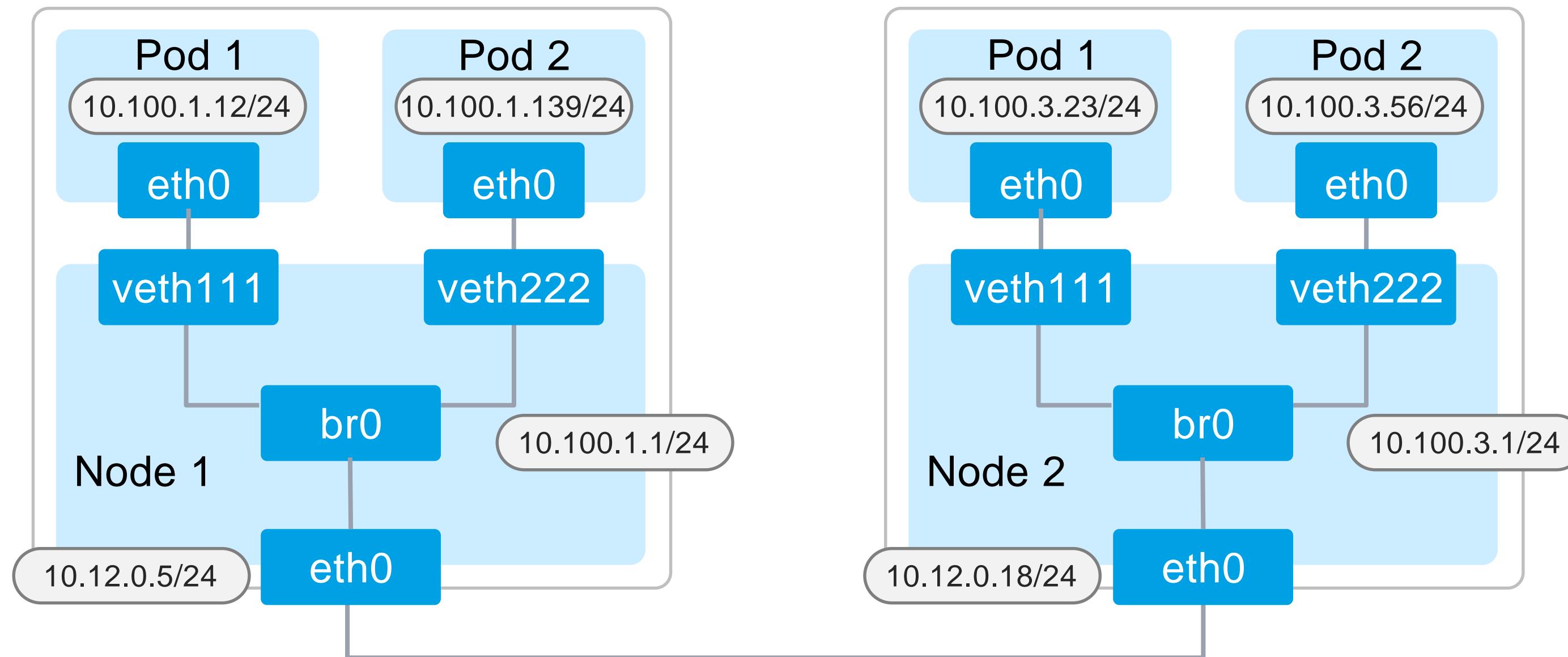
Network



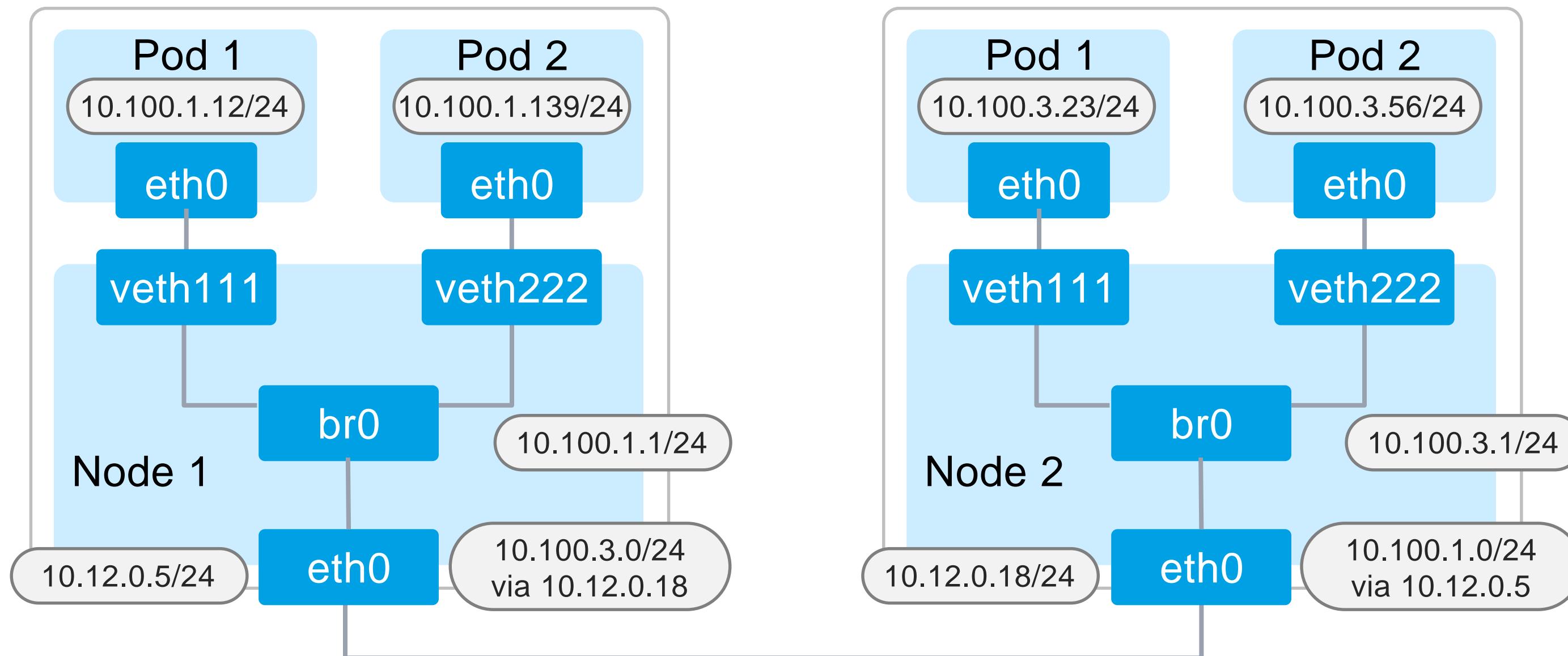
Flannel [host-gw]



Flannel [host-gw]

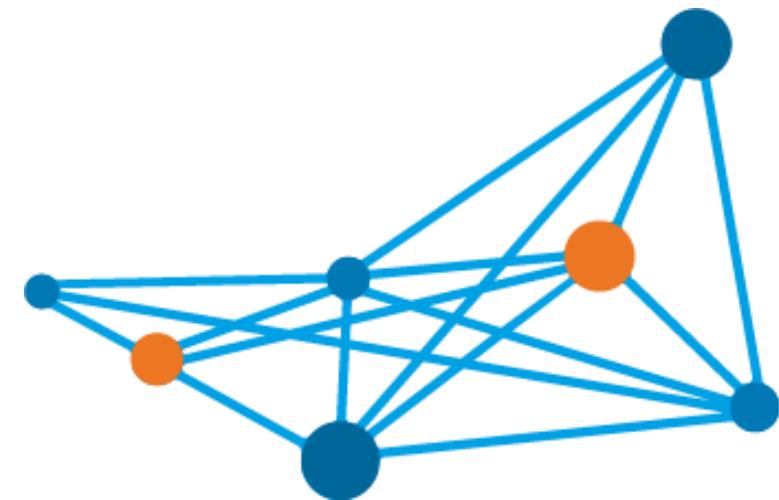


Flannel [host-gw]



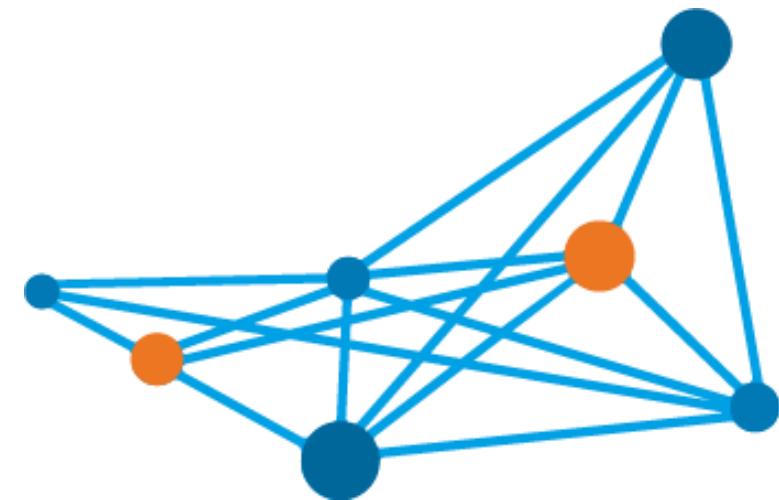
Network plugin (Flannel, Calico...)

- Обеспечивает связь между нодами и подами



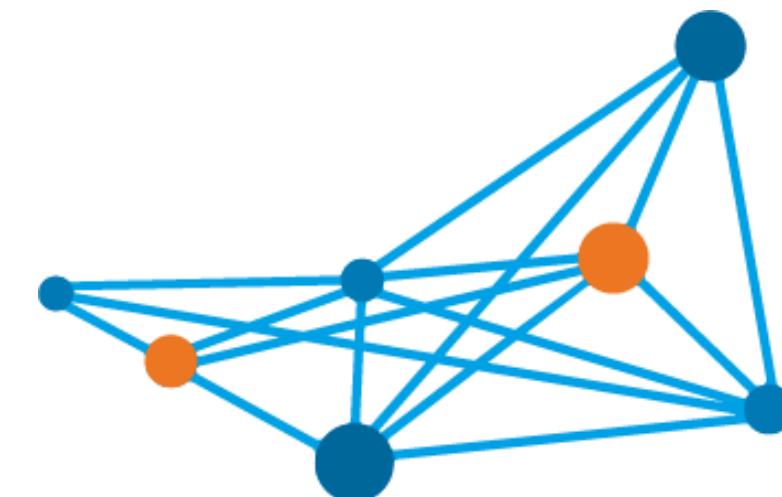
Network plugin (Flannel, Calico...)

- Обеспечивает связь между нодами и подами
- Раздает IP-адреса подам



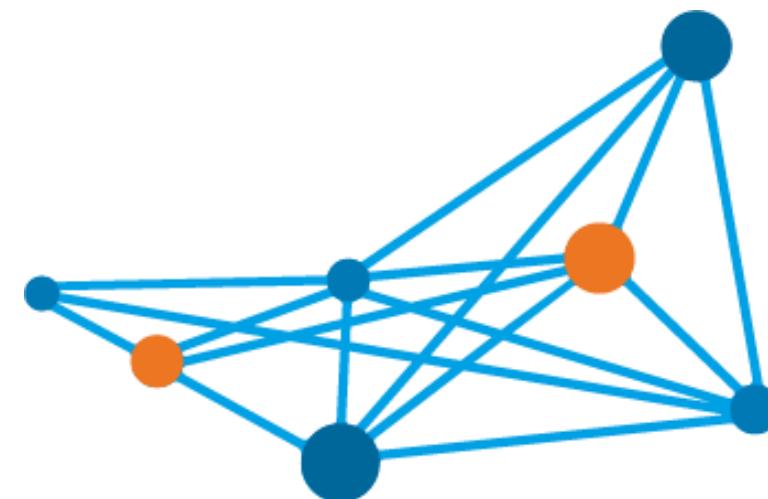
Network plugin (Flannel, Calico...)

- Обеспечивает связь между нодами и подами
- Раздает IP-адреса подам
- Реализует шифрование между нодами*



Network plugin (Flannel, Calico...)

- Обеспечивает связь между нодами и подами
- Раздает IP-адреса подам
- Реализует шифрование между нодами*
- Управляет Network Policies*



Ingress

```
$ kubectl exec --namespace kube-system nginx-ingress-controller-2536117660-1qmz3
cat /etc/nginx/nginx.conf

upstream mynamespace-service1-80 {
    # Load balance algorithm; empty for round robin, which is the default
    least_conn;

    keepalive 32;

    server 10.102.3.49:80 max_fails=0 fail_timeout=0;
    server 10.102.0.93:80 max_fails=0 fail_timeout=0;
}

}
```

Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: service1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: service2
          servicePort: 80
```

Ingress

```
$ kubectl exec --namespace kube-system nginx-ingress-controller-2536117660-1qmz3  
cat /etc/nginx/nginx.conf
```

```
server {  
    server_name foo.bar.com;  
    listen 80;  
    location / {  
        set $proxy_upstream_name "mynamespace-service1-80";  
        proxy_pass http://mynamespace-service1-80;  
    }  
}
```

Ingress

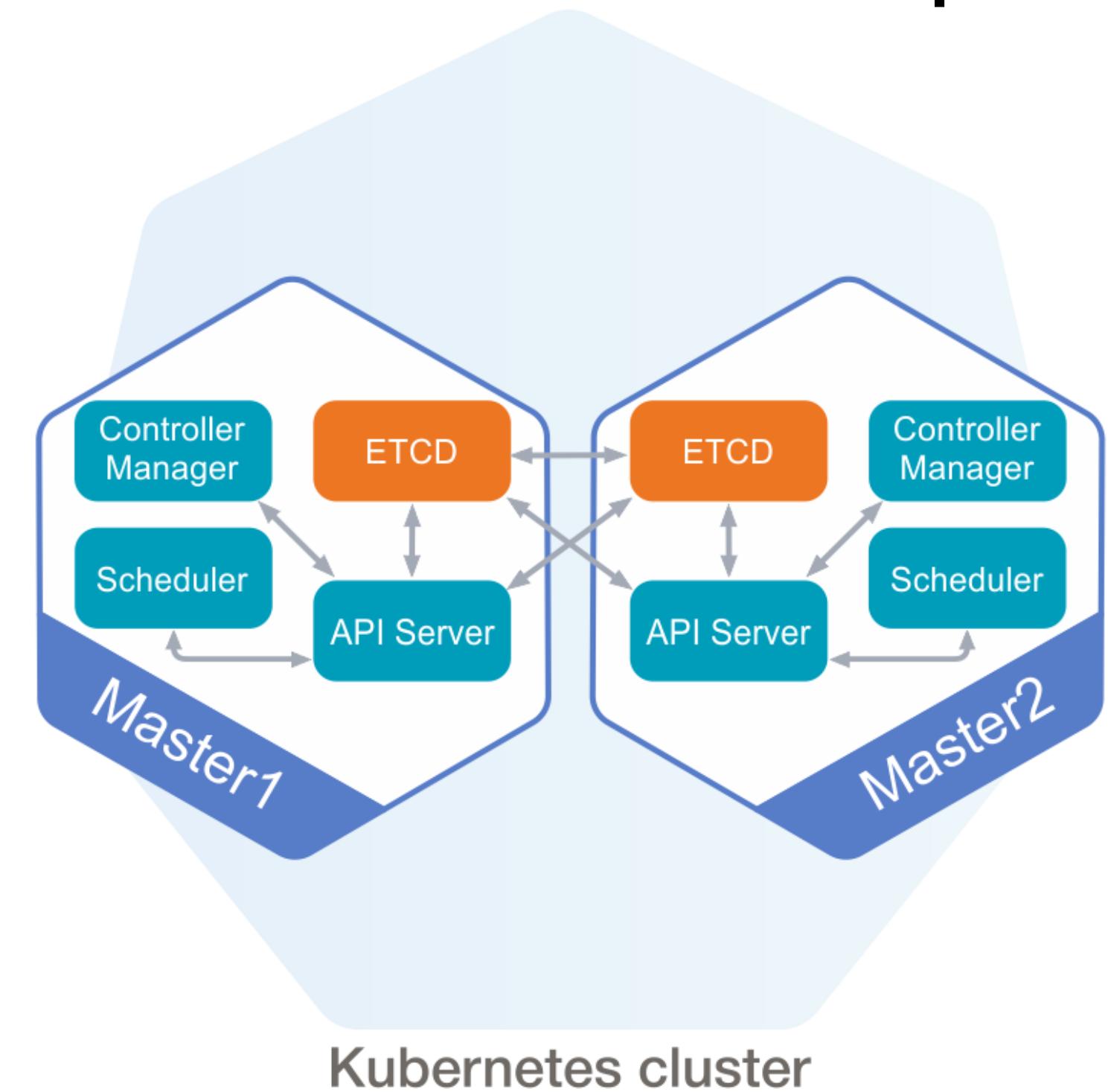
```
$ kubectl exec --namespace kube-system nginx-ingress-controller-2536117660-1qmz3  
cat /etc/nginx/nginx.conf
```

```
log_format upstreaminfo '$the_real_ip - [$the_real_ip] - $remote_user [$time_local]  
"$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"  
$request_length $request_time [$proxy_upstream_name] $upstream_addr  
$upstream_response_length $upstream_response_time $upstream_status';
```

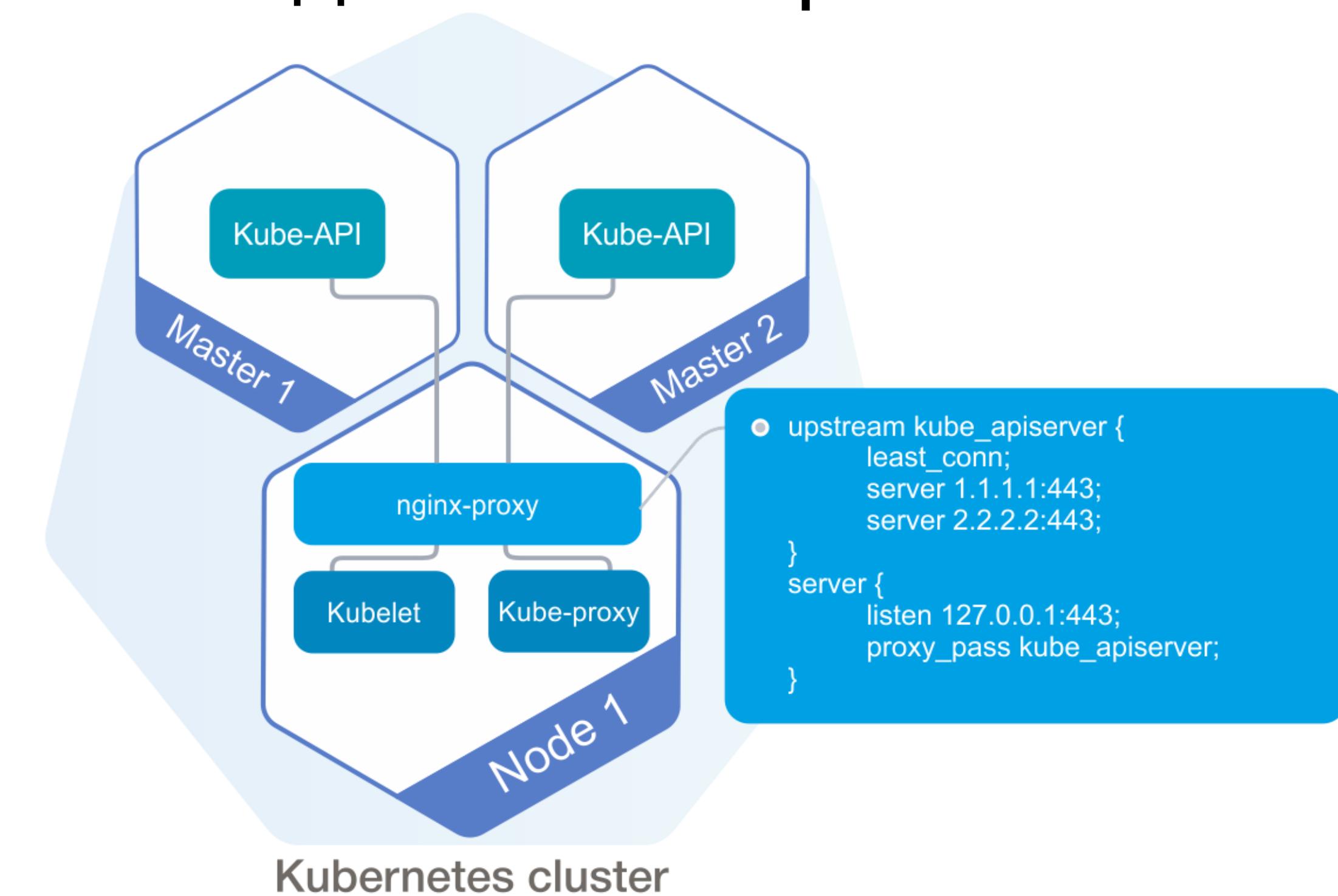
It's **NOT** magic



Отказоустойчивый сетап мастеров



Взаимодействие ноды -> мастера



Что мы узнали?

- Какие есть компоненты кластера и как они взаимодействуют
- Что такое Service и как он реализуется
- Как организована сеть внутри кластера
- Что из себя представляет Ingress и его контроллер
- Как обеспечивается отказоустойчивость внутри кластера

СЛЁРМ



Southbridge

Перерыв