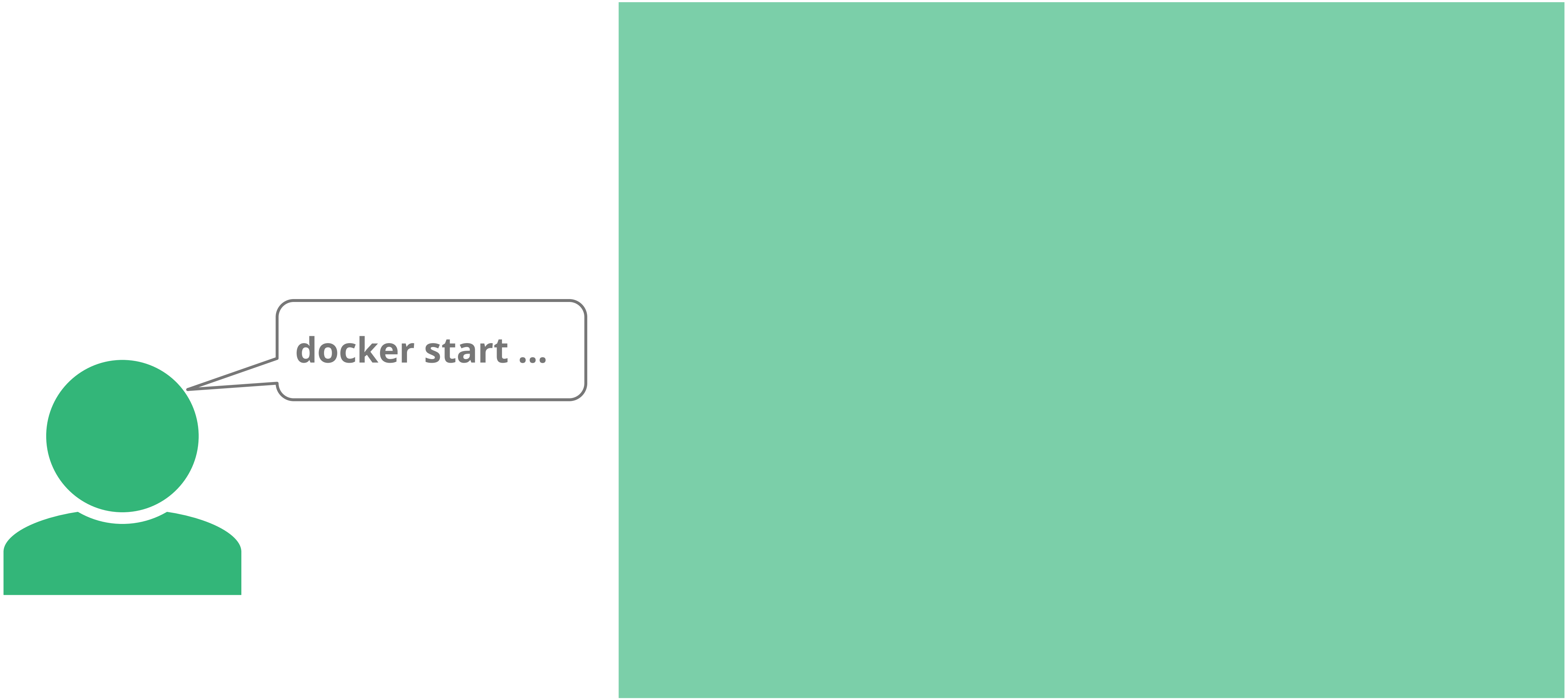
Kubernetes Networking

Seattle Kubernetes Meetup

CJ Cullen <cjcullen@google.com>
Software Engineer
@cj_cullen
github.com/cjcullen

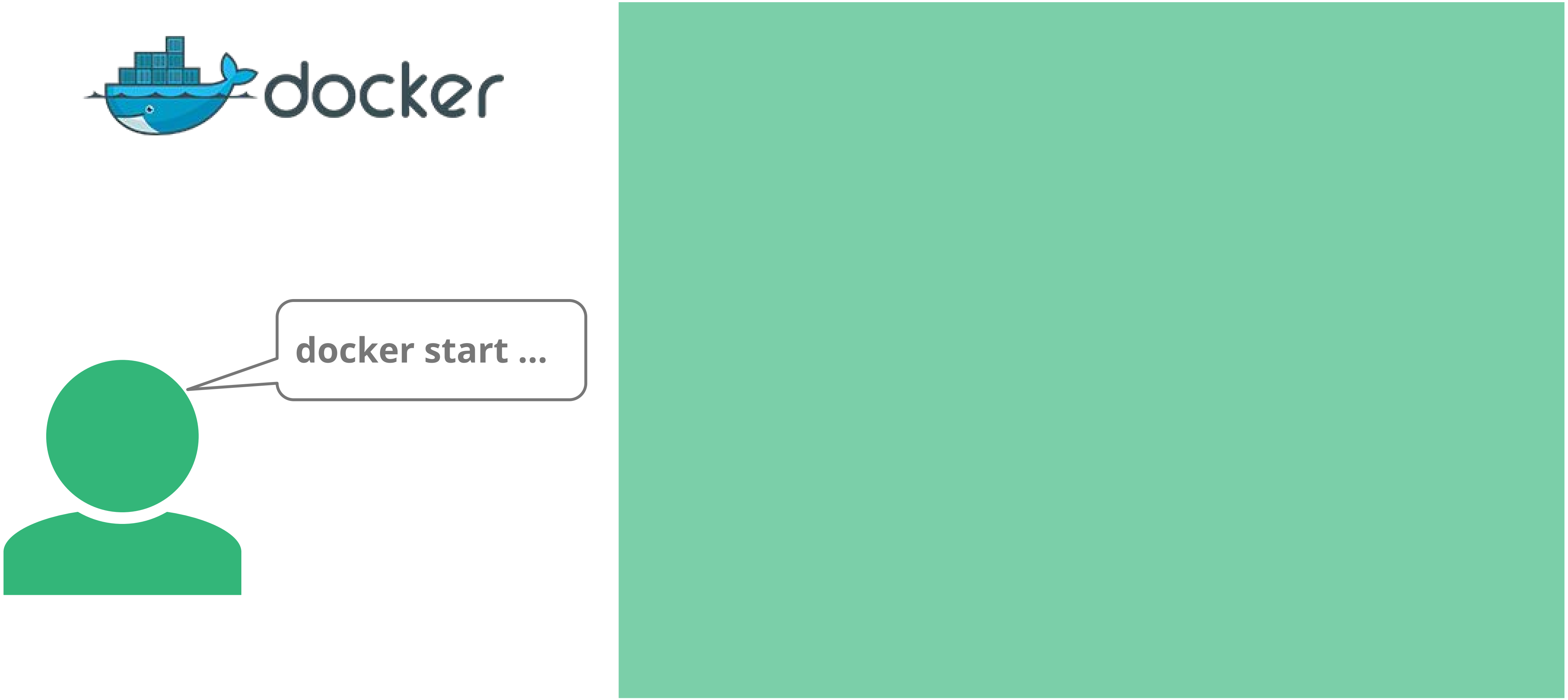
Docker Networking

Docker networking

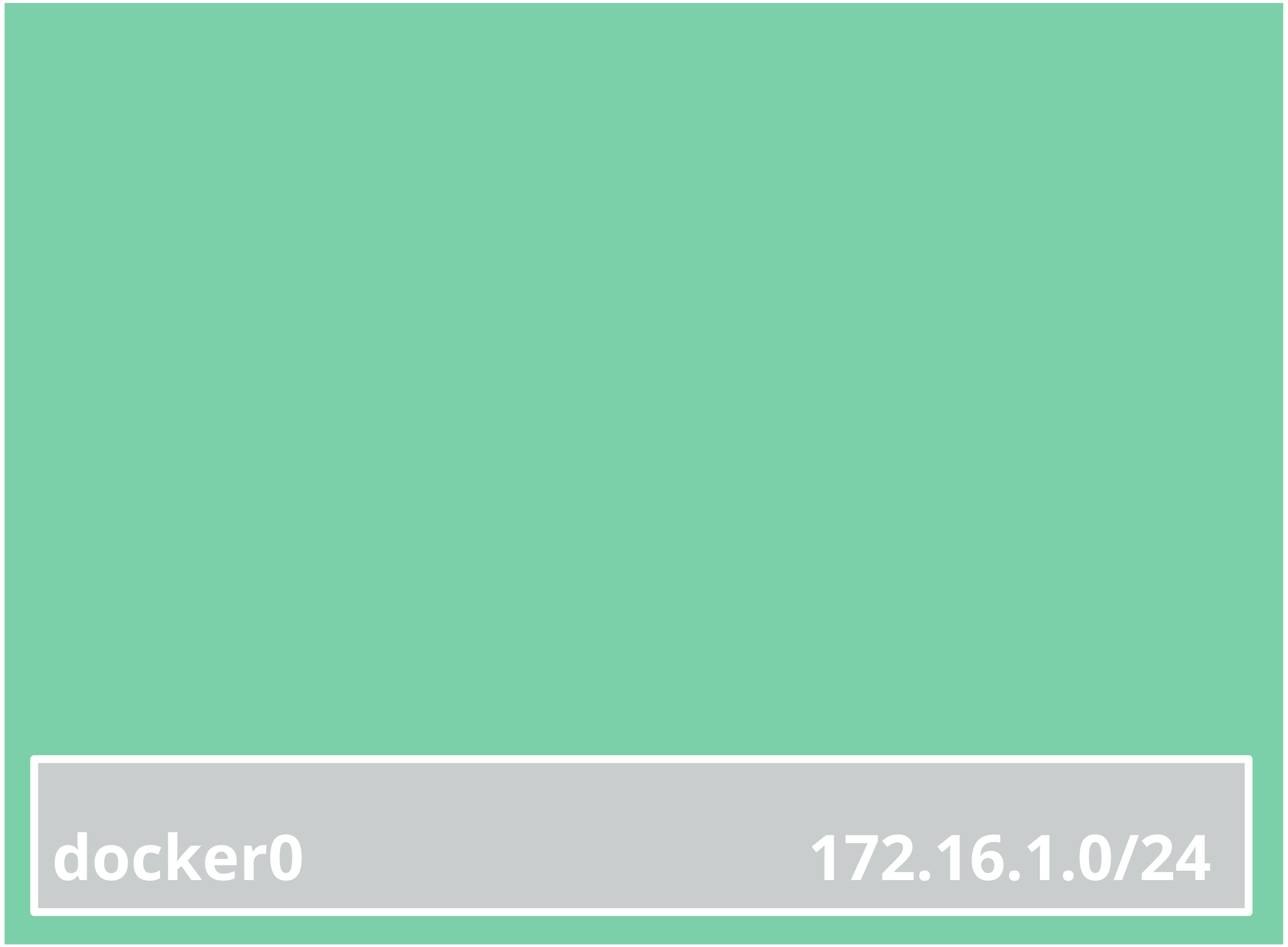


docker start ...

Docker networking



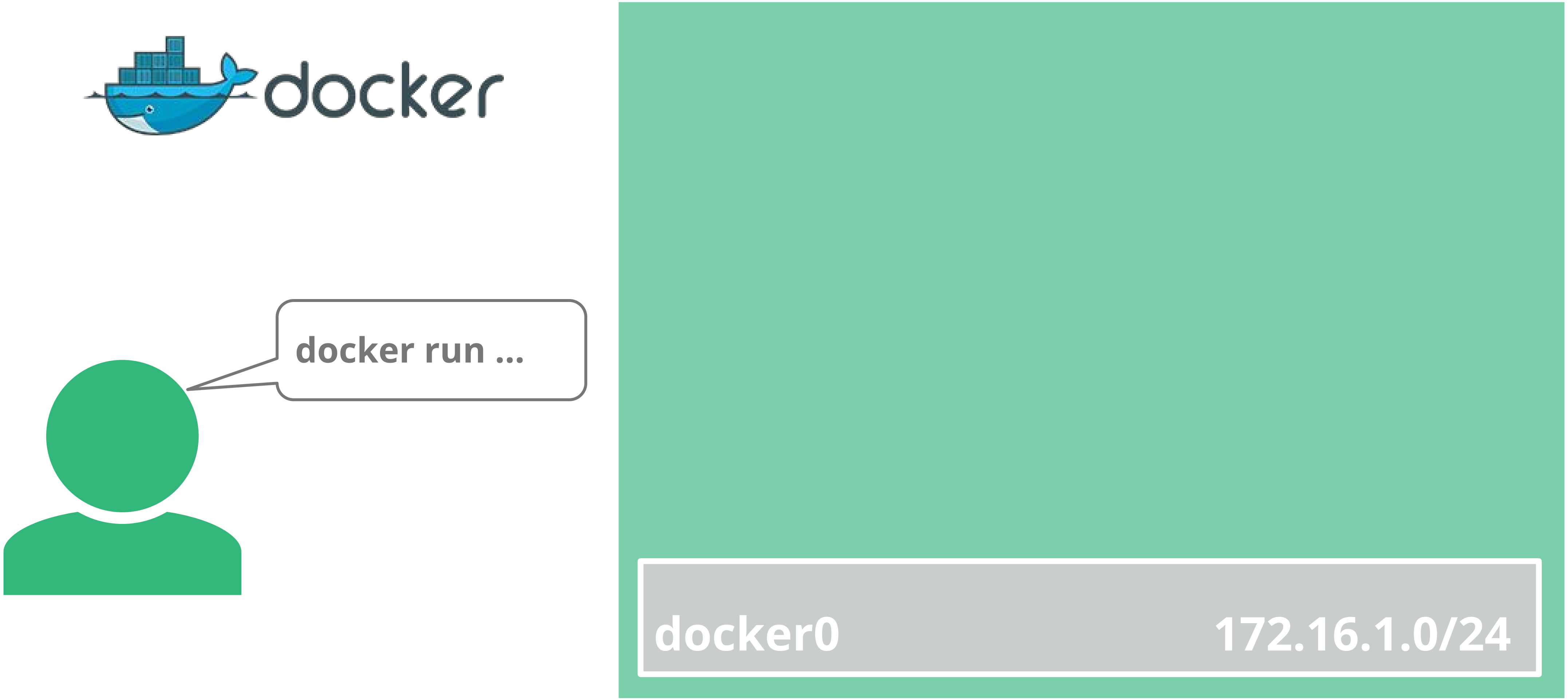
Docker networking



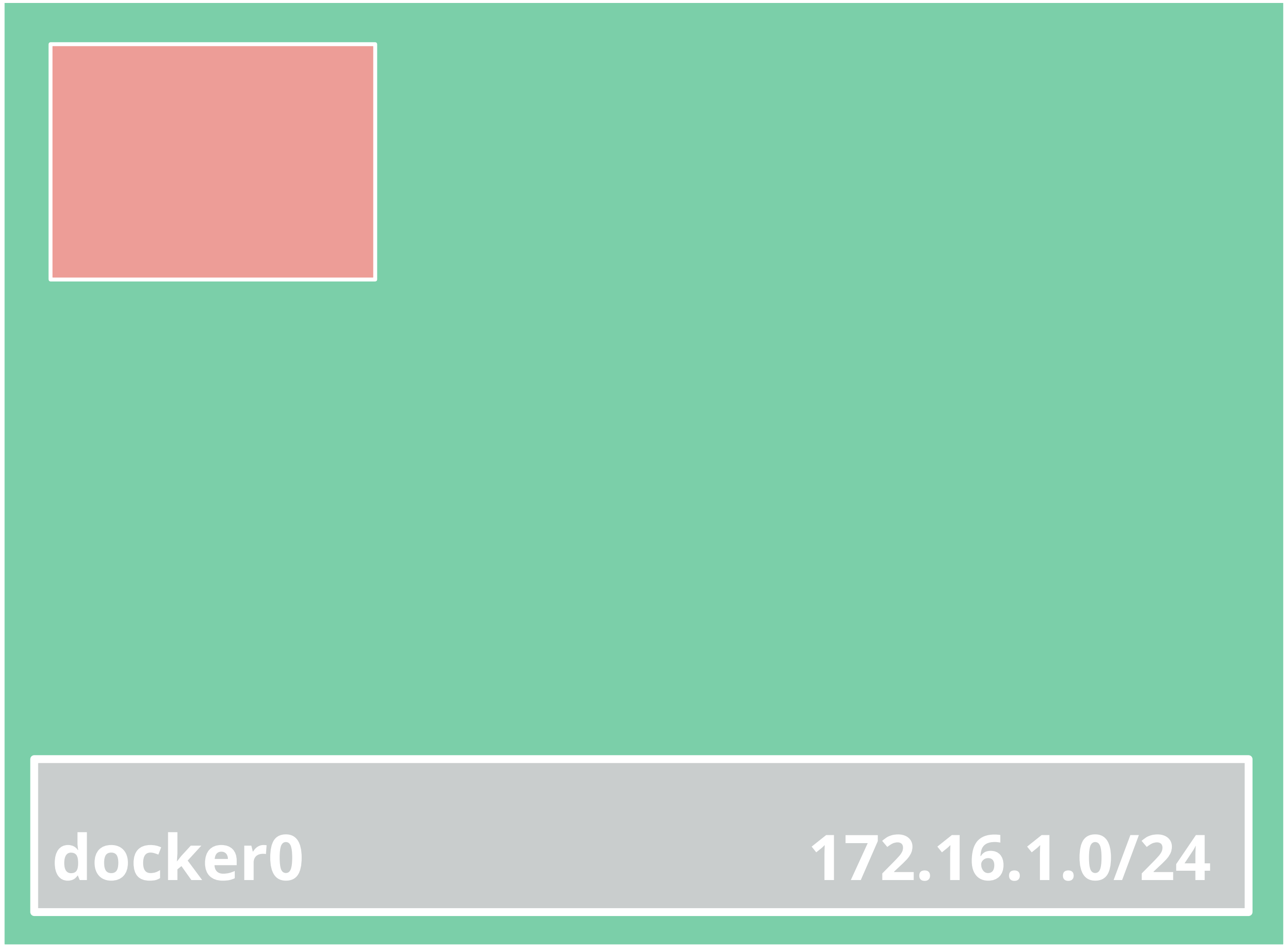
docker0

172.16.1.0/24

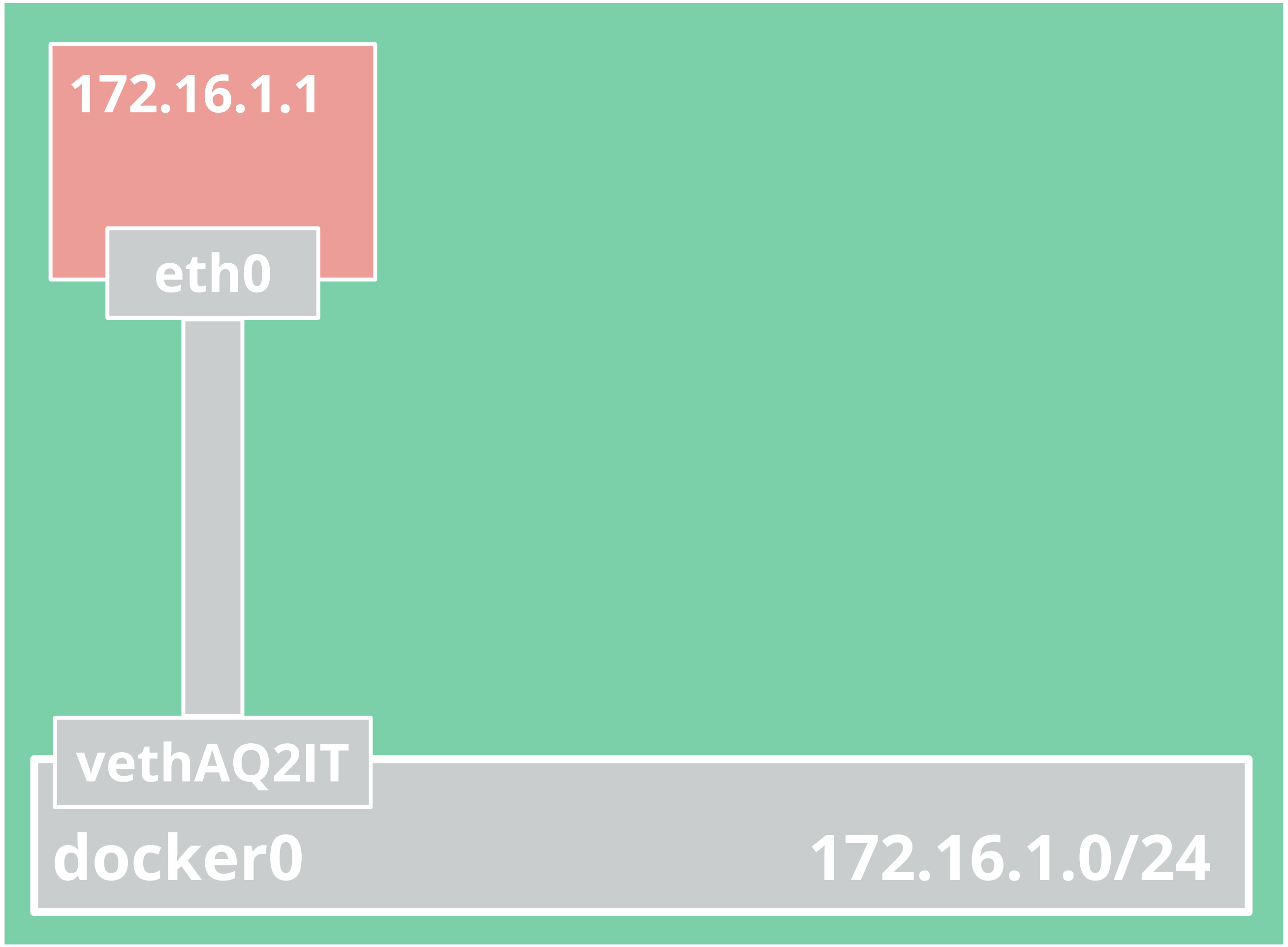
Docker networking



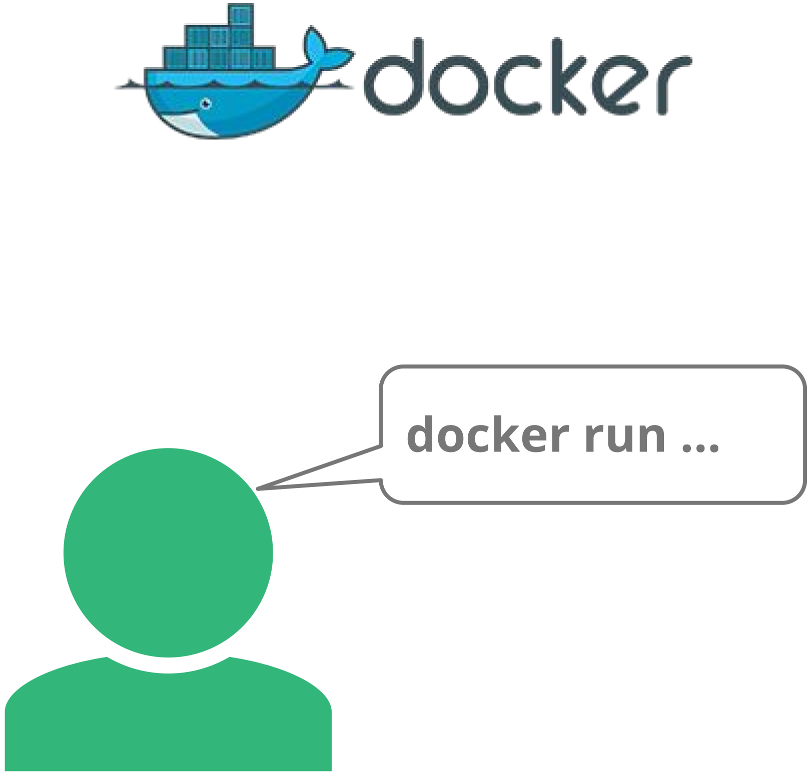
Docker networking

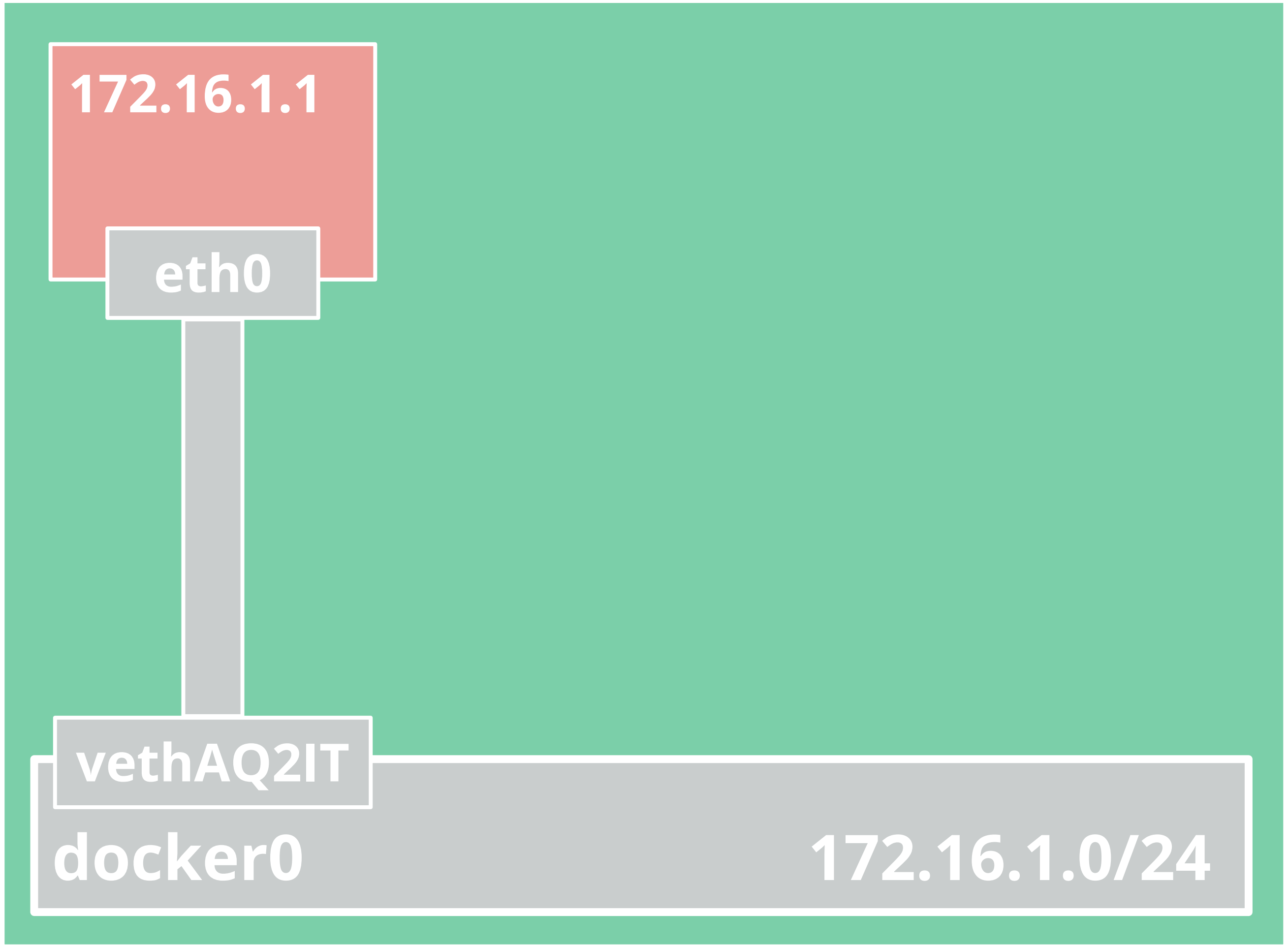


Docker networking

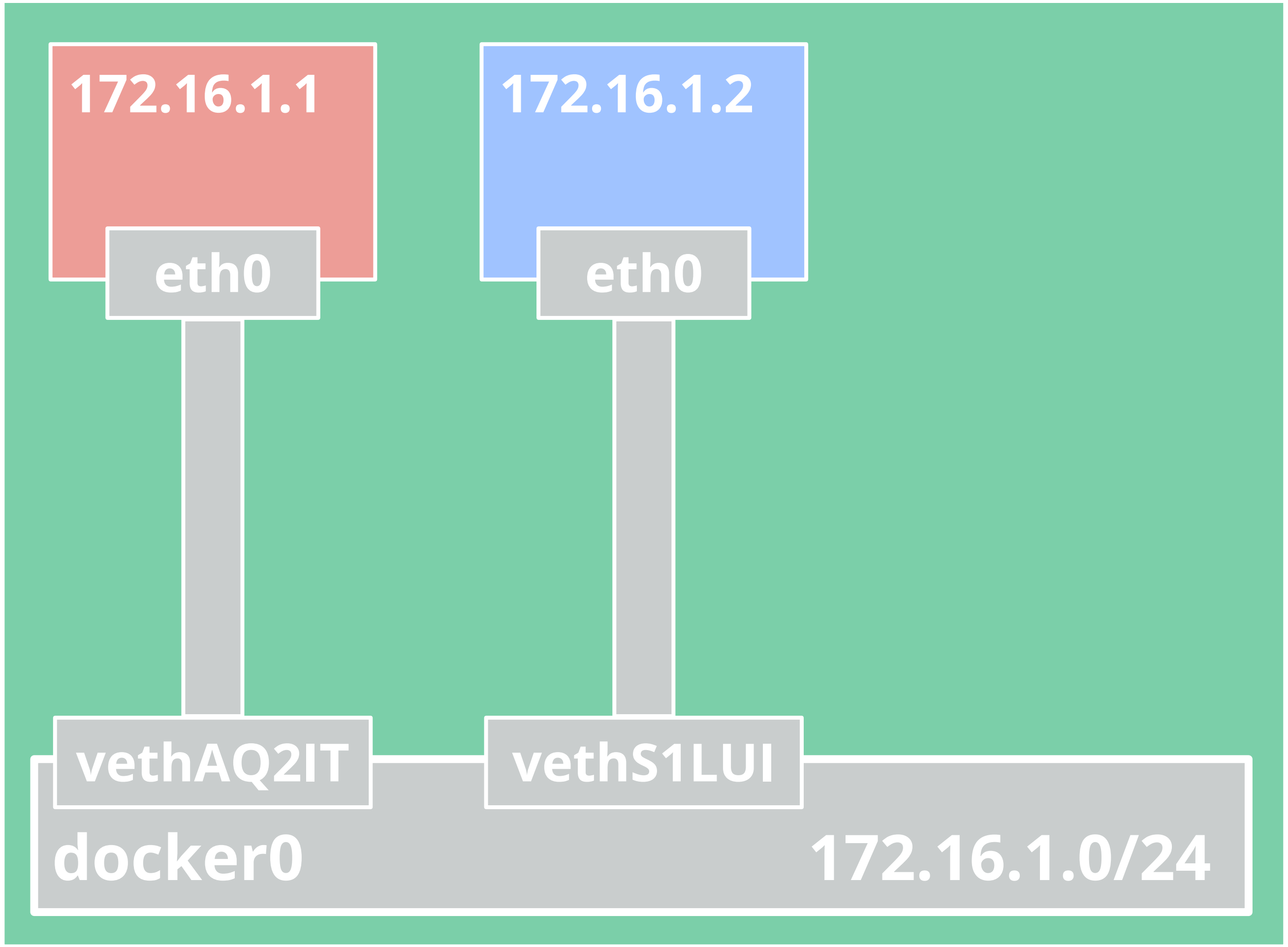


Docker networking

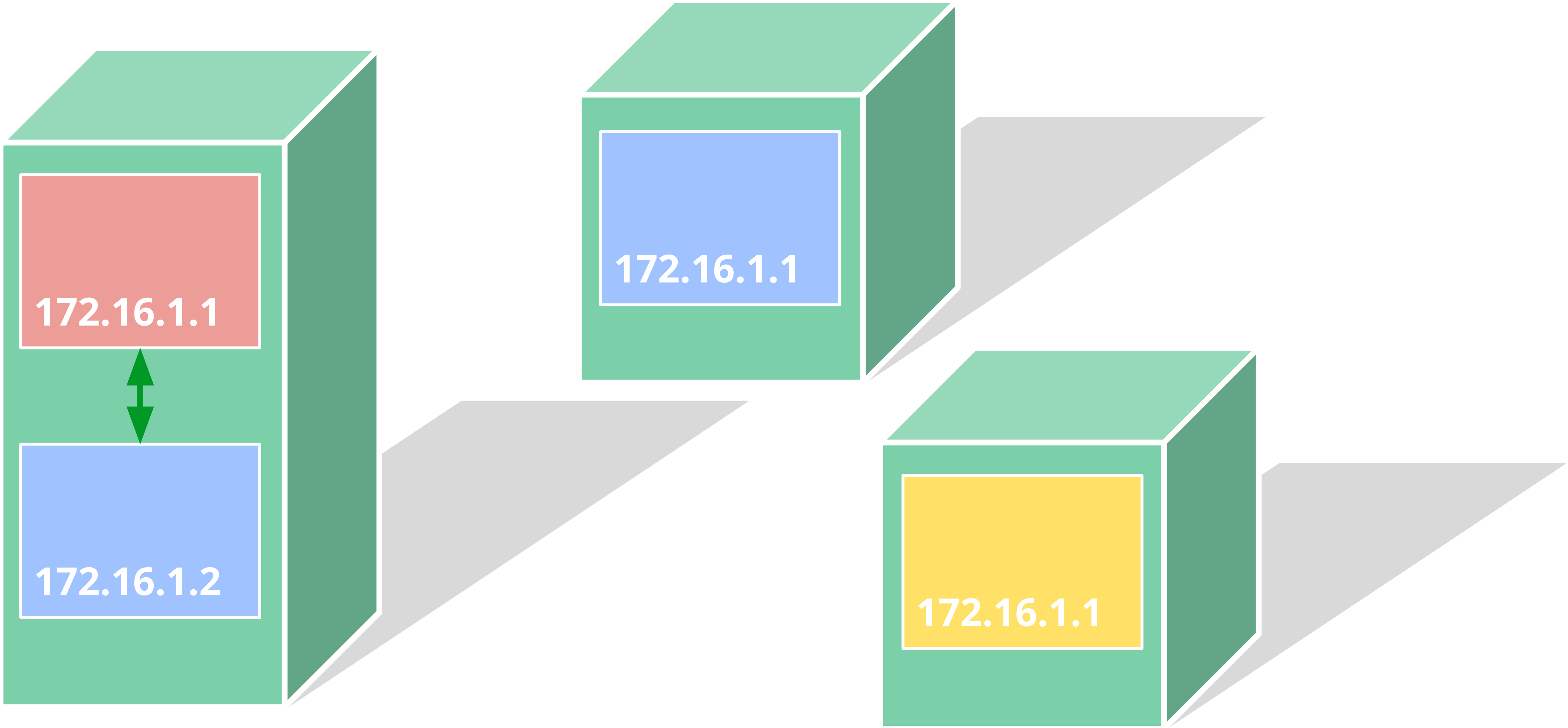




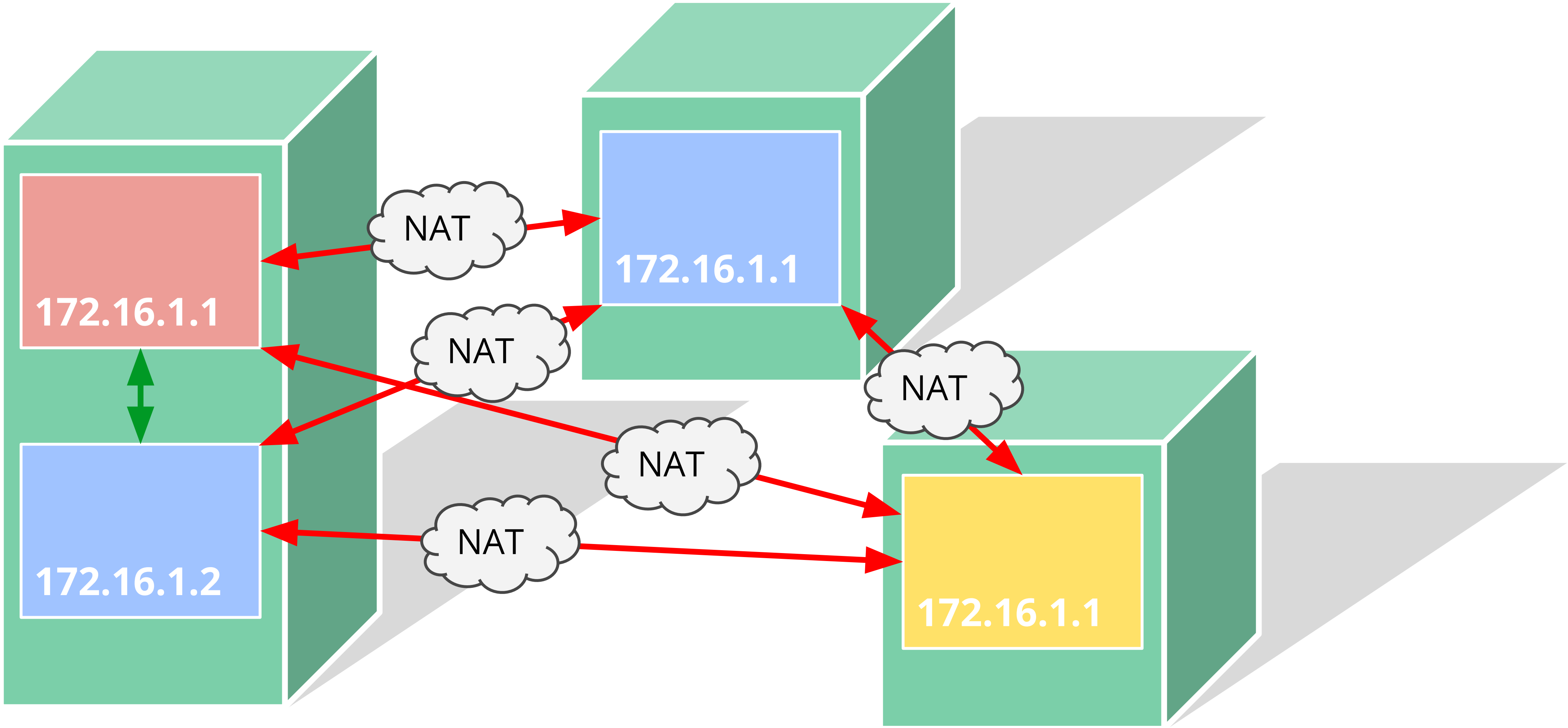
Docker networking



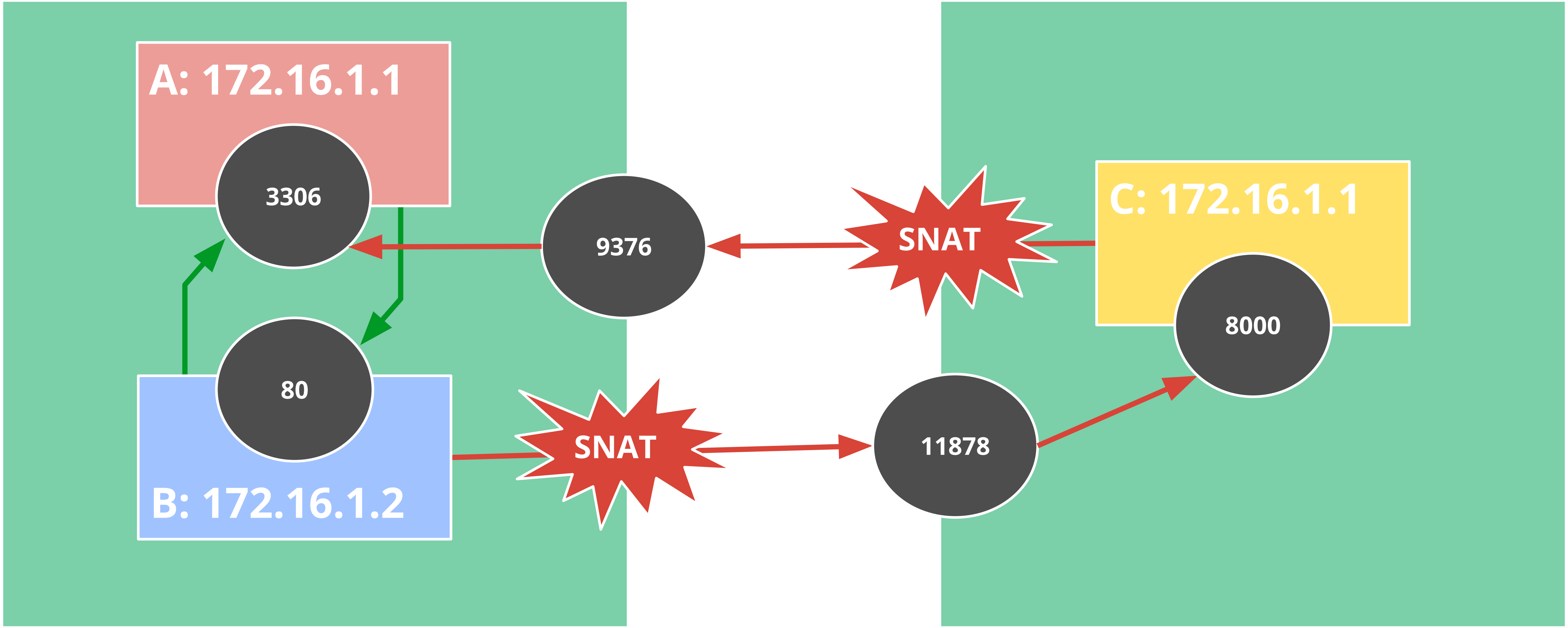
Docker networking



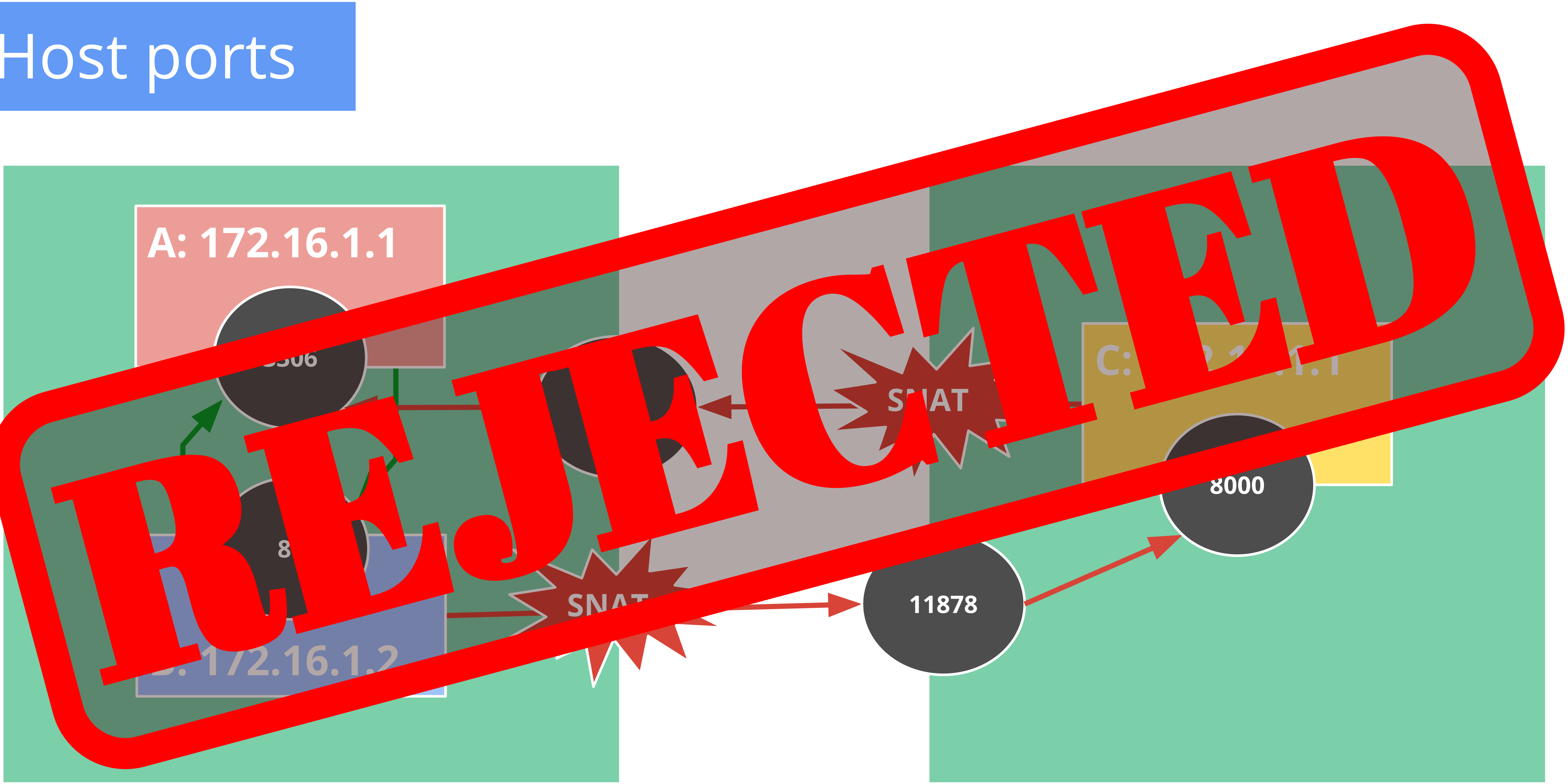
Docker networking



Host ports


A: 172.16.1.1
3306
9376
SNAT
C: 172.16.1.1
8000
80
SNAT
11878
B: 172.16.1.2

Host ports



Kubernetes Networking

Kubernetes networking

IPs are **routable**

- vs docker default private IP

Pods can reach each other without NAT

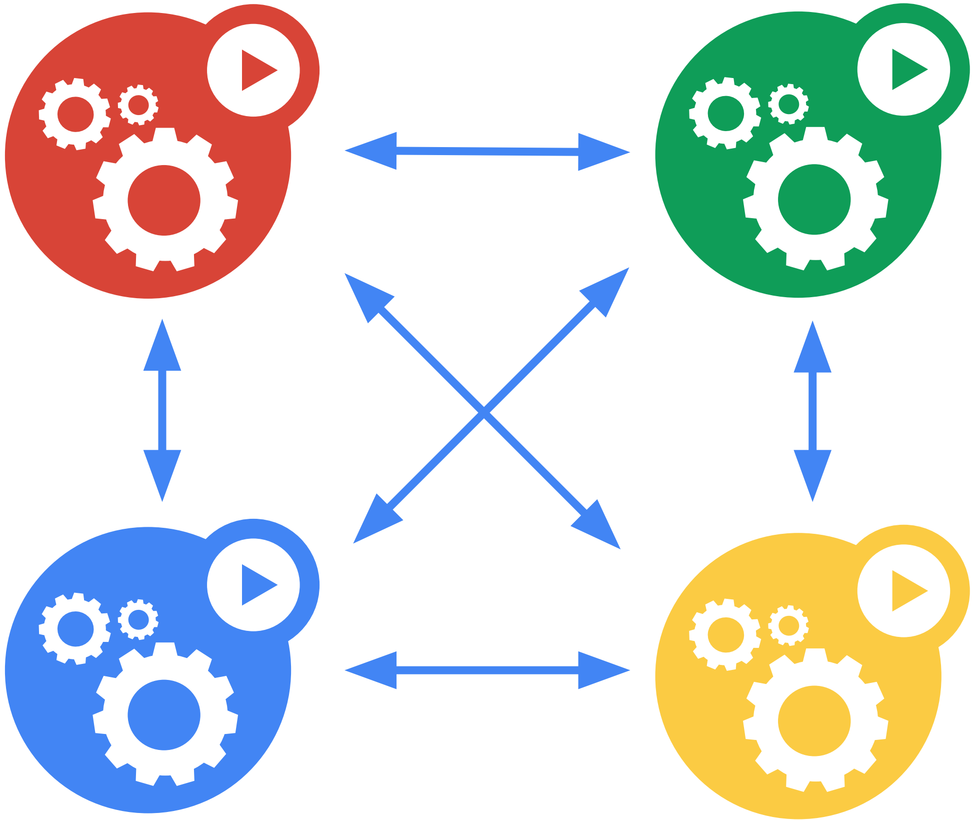
- even across nodes

No brokering of port numbers

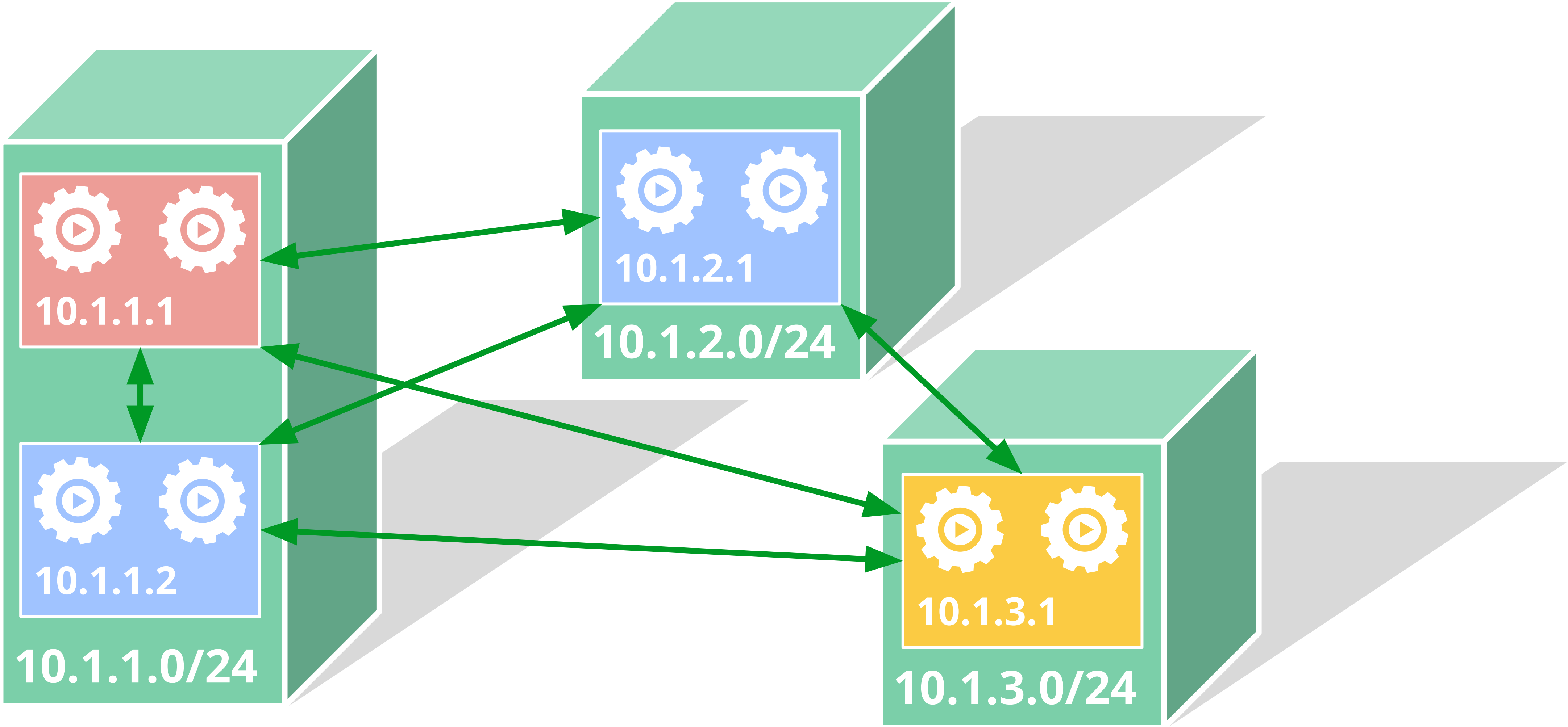
- too complex, why bother?

This is a fundamental requirement

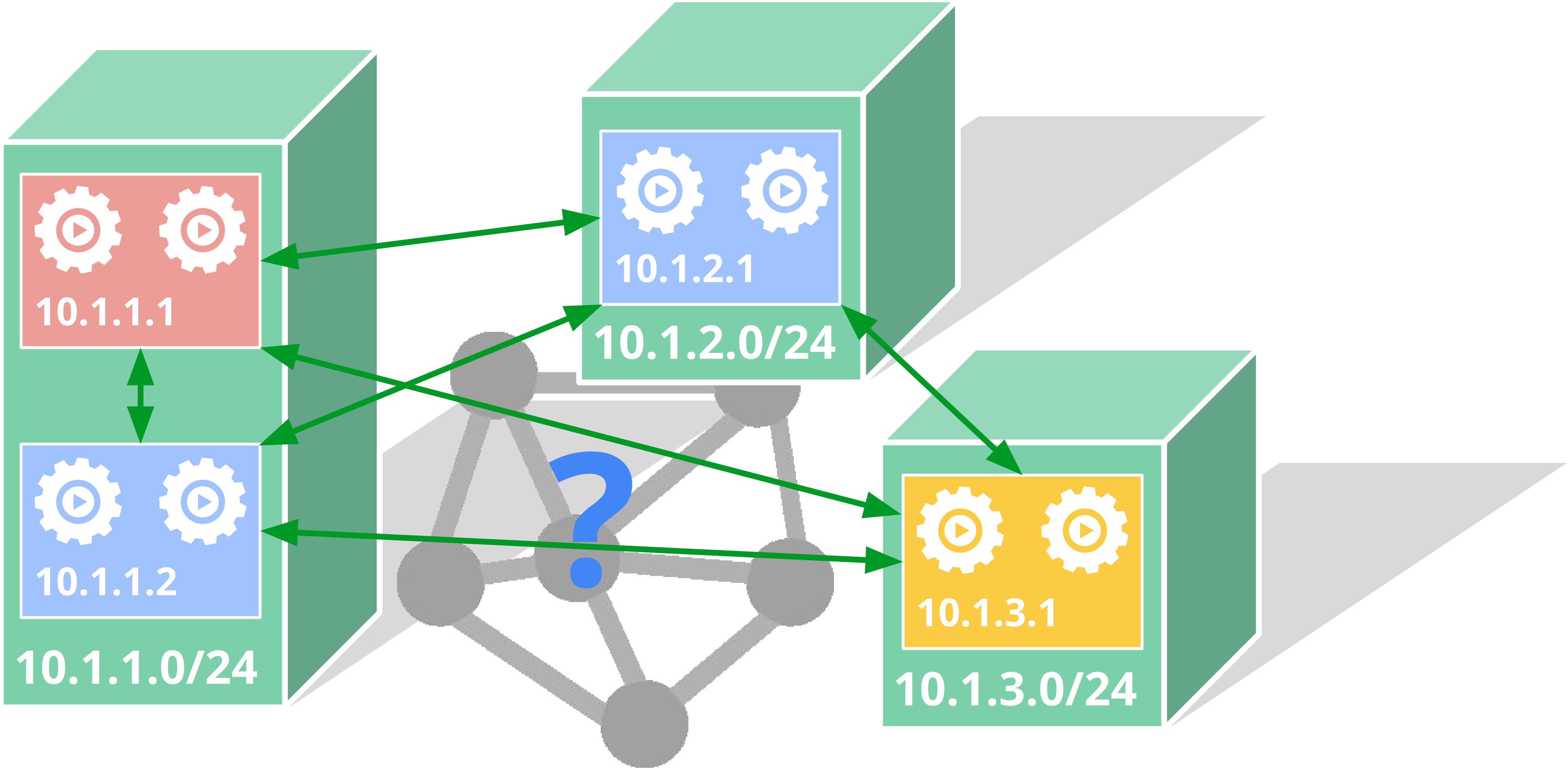
- can be L3 routed
- can be underlayed (cloud)
- can be overlayed (SDN)



Kubernetes networking



Kubernetes networking



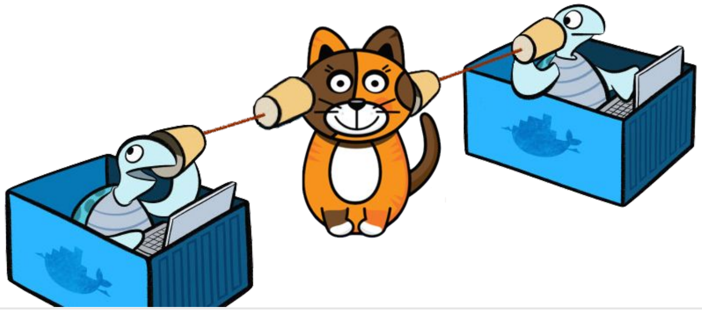
Kubernetes networking

On GCE/GKE

- GCE Advanced Routes (program the fabric)
- “Everything to 10.1.1.0/24, send to this VM”

Plenty of other ways

- AWS: Route Tables
- Weave
- Calico
- Flannel
- OVS
- OpenContrail
- Cisco Contiv
- Others…



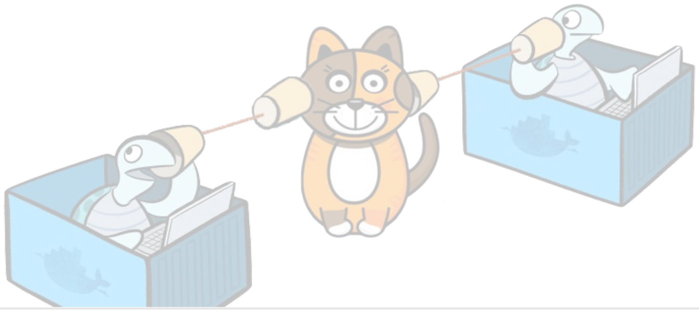
Kubernetes networking

On **GCE/GKE**

- GCE Advanced Routes (program the fabric)
- “Everything to 10.1.1.0/24, send to this VM”

Plenty of other ways

- **AWS**: Route Tables
- Weave
- Calico
- Flannel
- OVS
- OpenContrail
- Cisco Contiv
- Others...



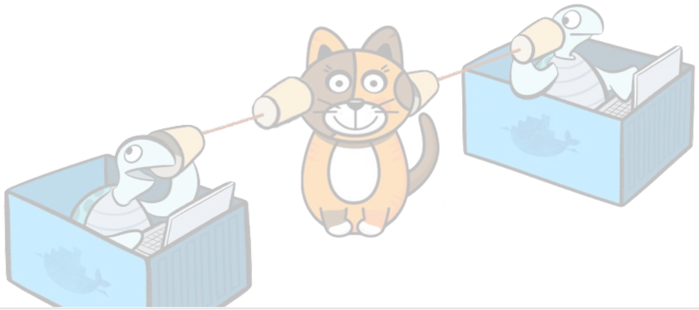
Kubernetes networking

On GCE/GKE

- GCE Advanced Routes (program the fabric)
- “Everything to 10.1.1.0/24, send to this VM”

Plenty of other ways

- AWS: Route Tables
- Weave
- Calico
- **Flannel**
- OVS
- OpenContrail
- Cisco Contiv
- Others...



Pods

Pods

Small group of containers & volumes

Tightly coupled

The atom of scheduling & placement

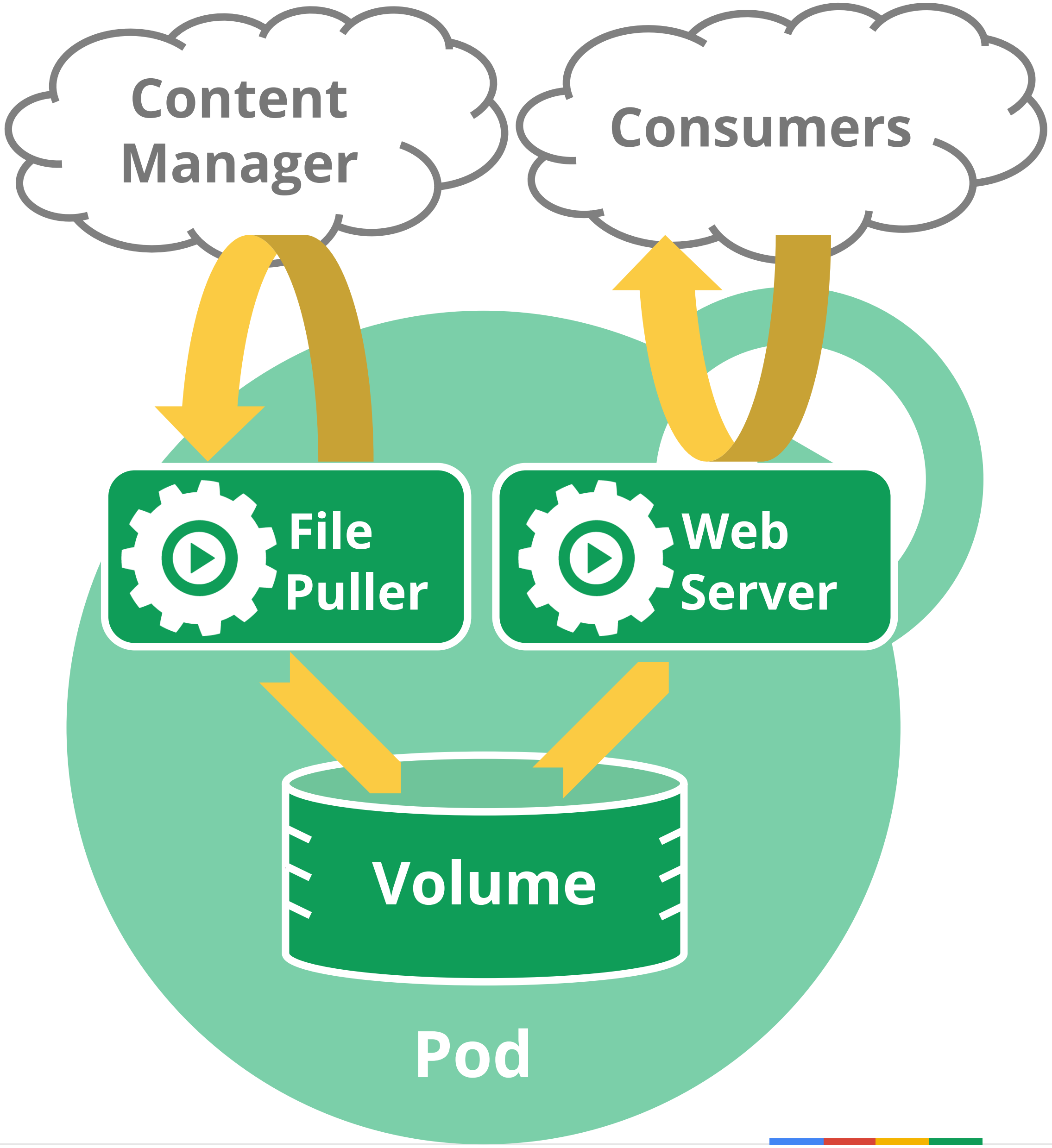
Shared namespace

- share IP address & localhost
- share IPC, etc.

Managed lifecycle

- bound to a node, restart in place
- can die, cannot be reborn with same ID

Example: data puller & web server



Pods

Small group of containers & volumes

Tightly coupled

The atom of scheduling & placement

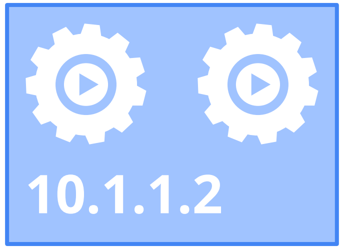
Shared namespace

- share IP address & localhost
- share IPC, etc.

Managed lifecycle

- bound to a node, restart in place
- can die, cannot be reborn with same ID

Example: data puller & web server



Pods

Small group of containers & volumes

Tightly coupled

The atom of scheduling & placement

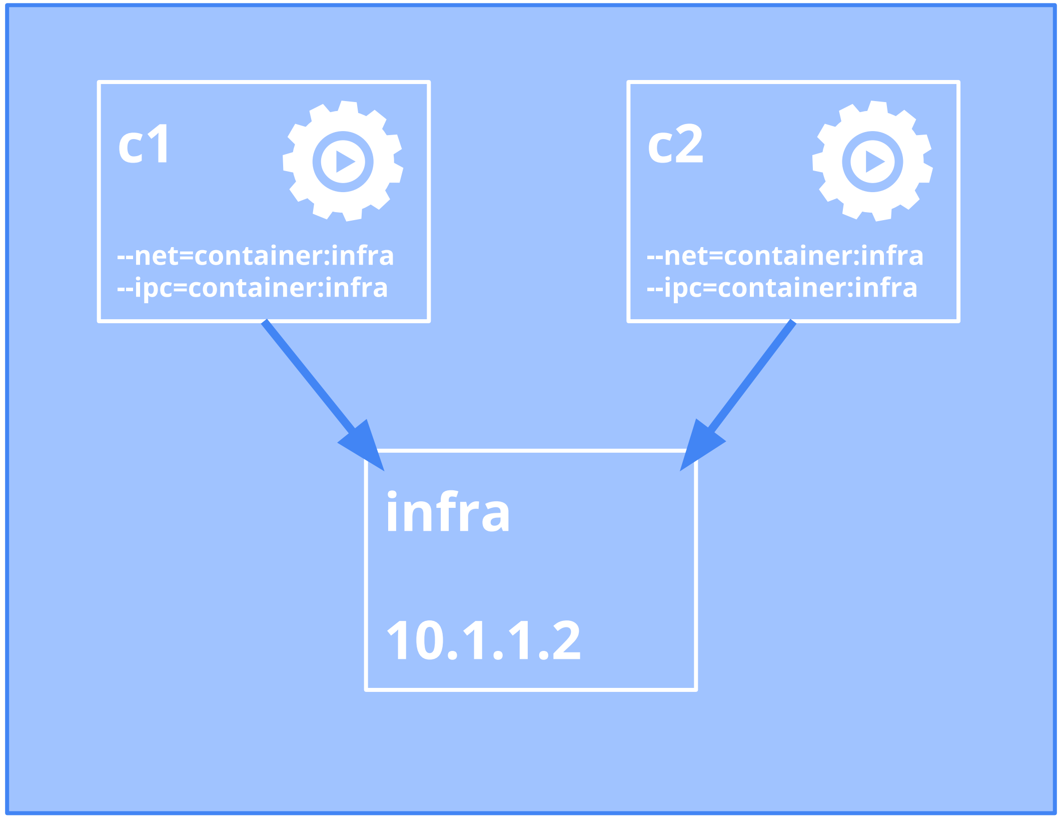
Shared namespace

- **share IP address & localhost**
- **share IPC, etc.**

Managed lifecycle

- bound to a node, restart in place
- can die, cannot be reborn with same ID

Example: data puller & web server



Services

Services

A group of pods that **work together**

- grouped by a selector

Defines access policy

- “load balanced” or “headless”

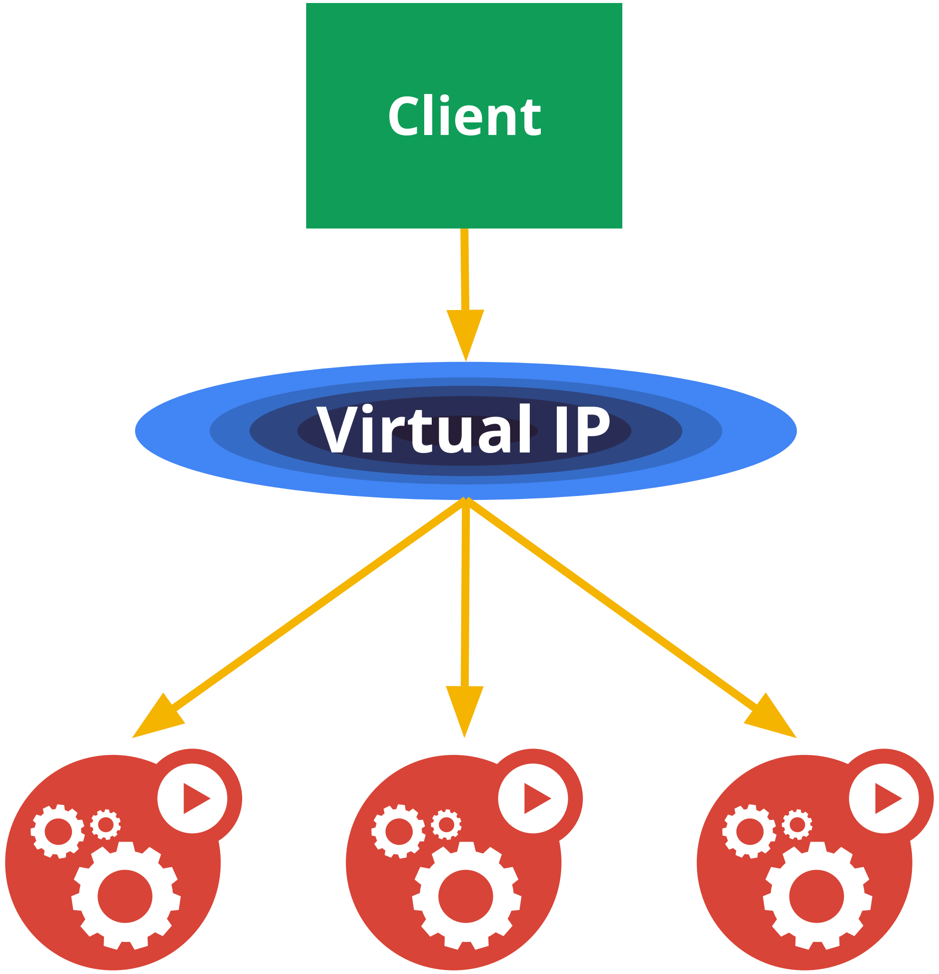
Gets a stable **virtual IP** and port

- sometimes called the service *portal*
- also a DNS name

VIP is managed by *kube-proxy*

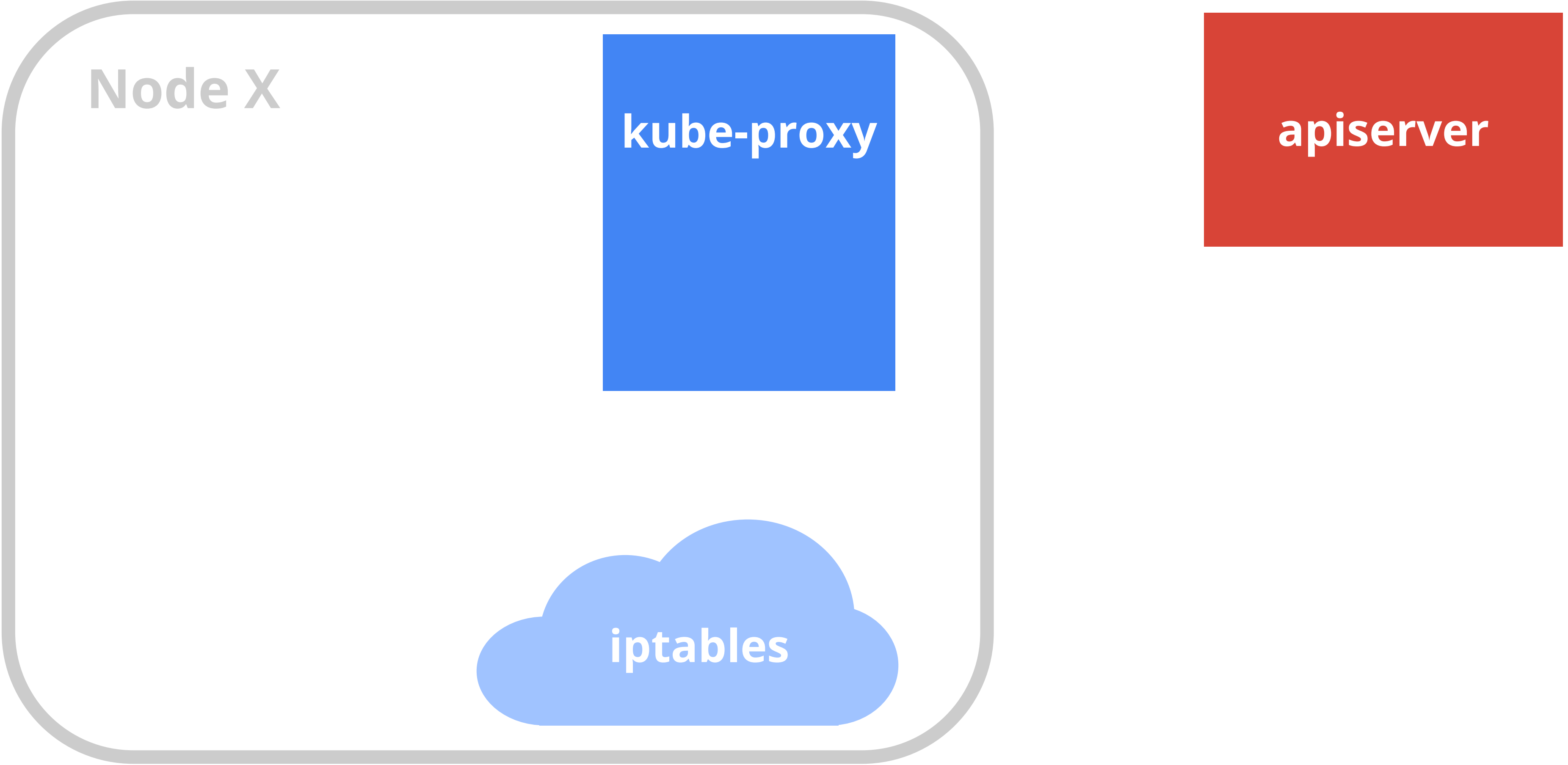
- watches all services
- updates iptables when backends change

Hides complexity - ideal for non-native apps

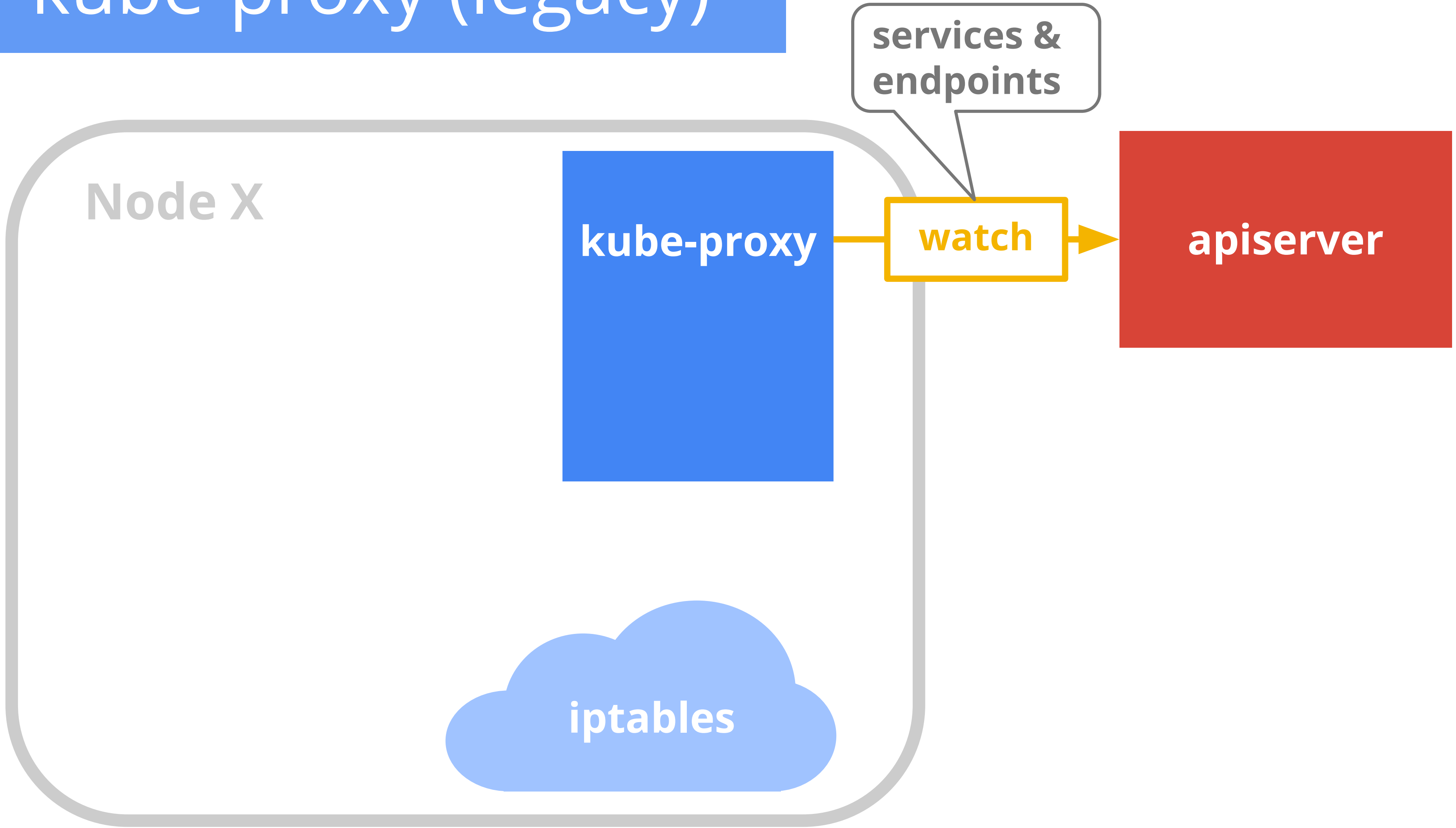


kube-proxy

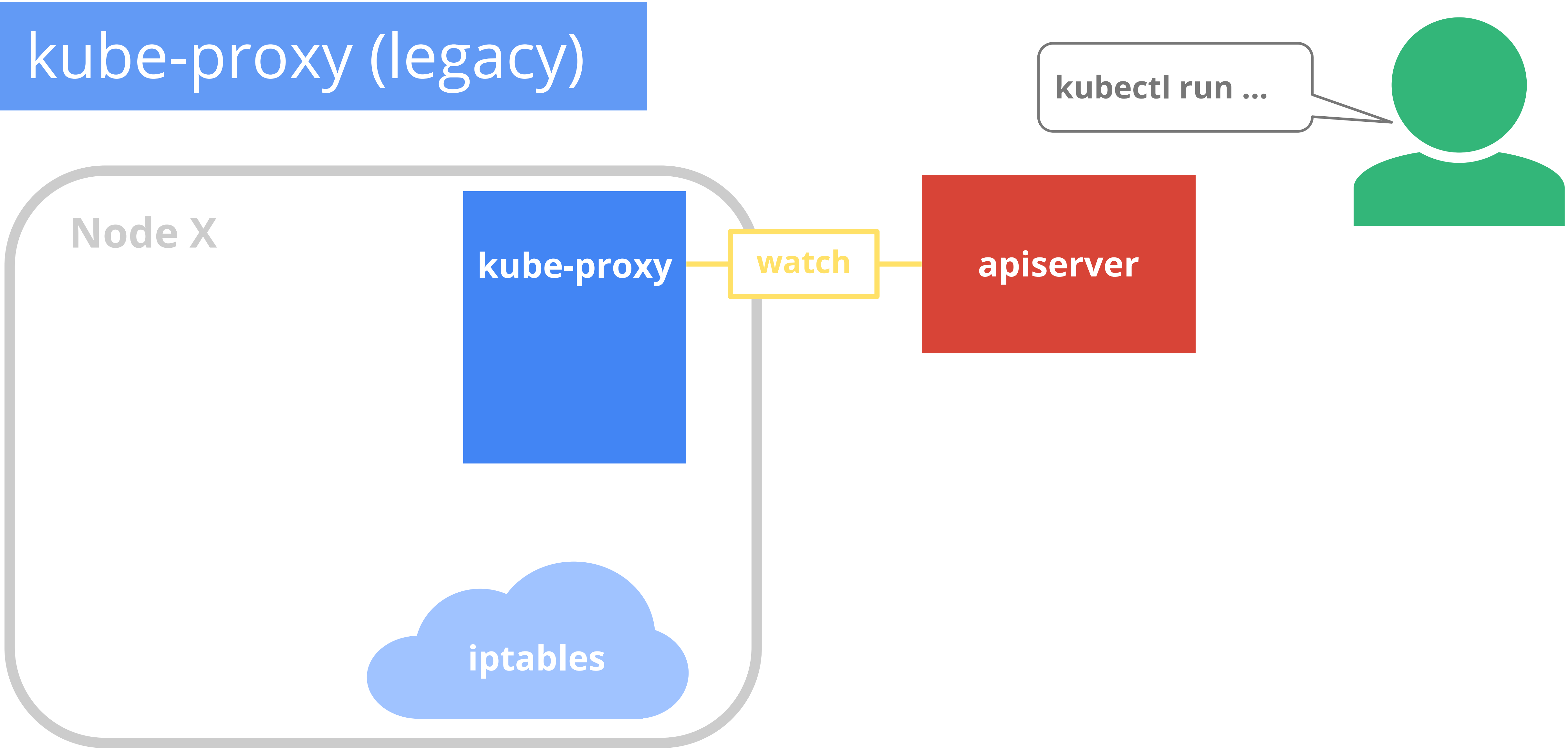
kube-proxy (legacy)



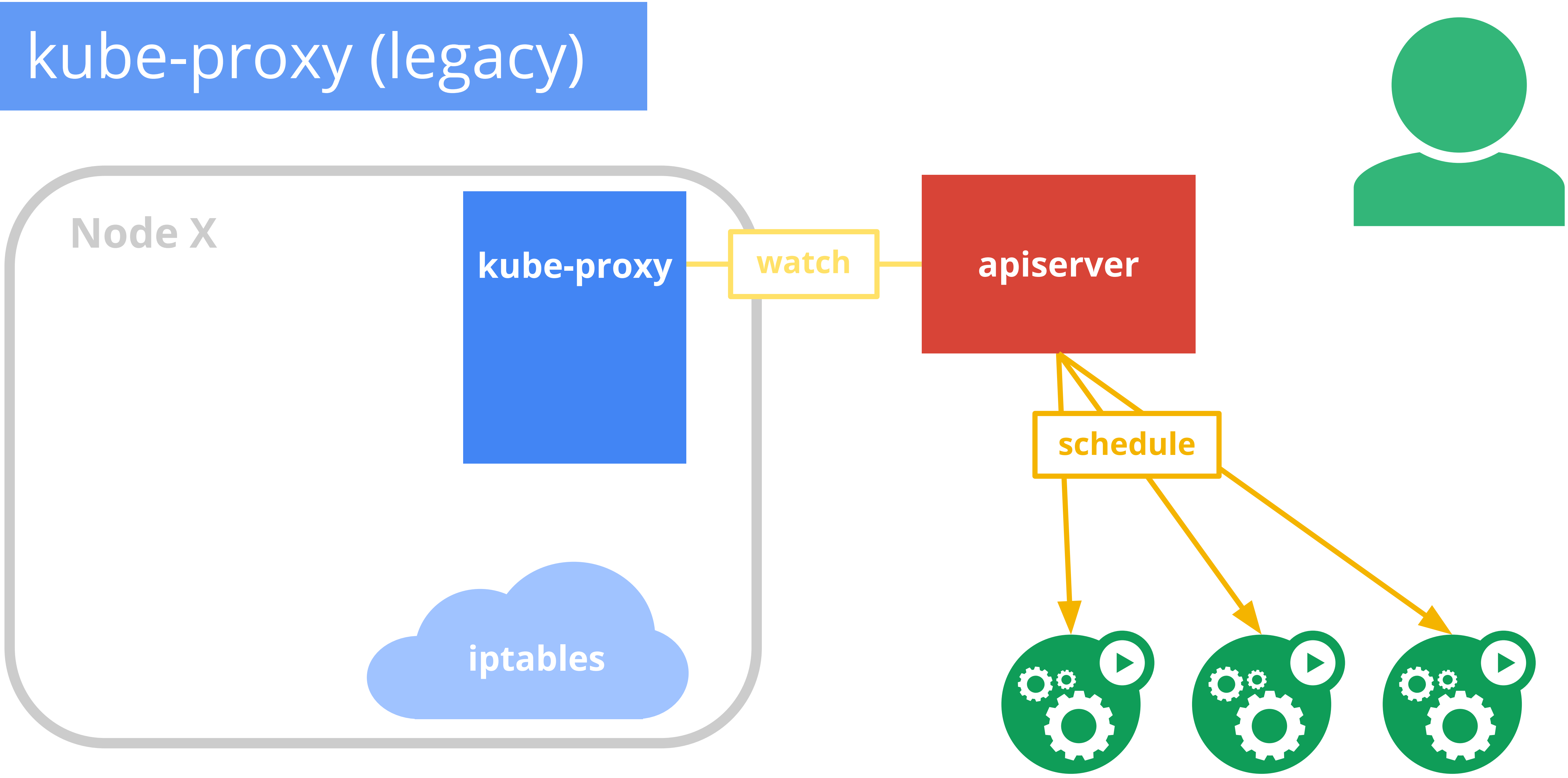
kube-proxy (legacy)



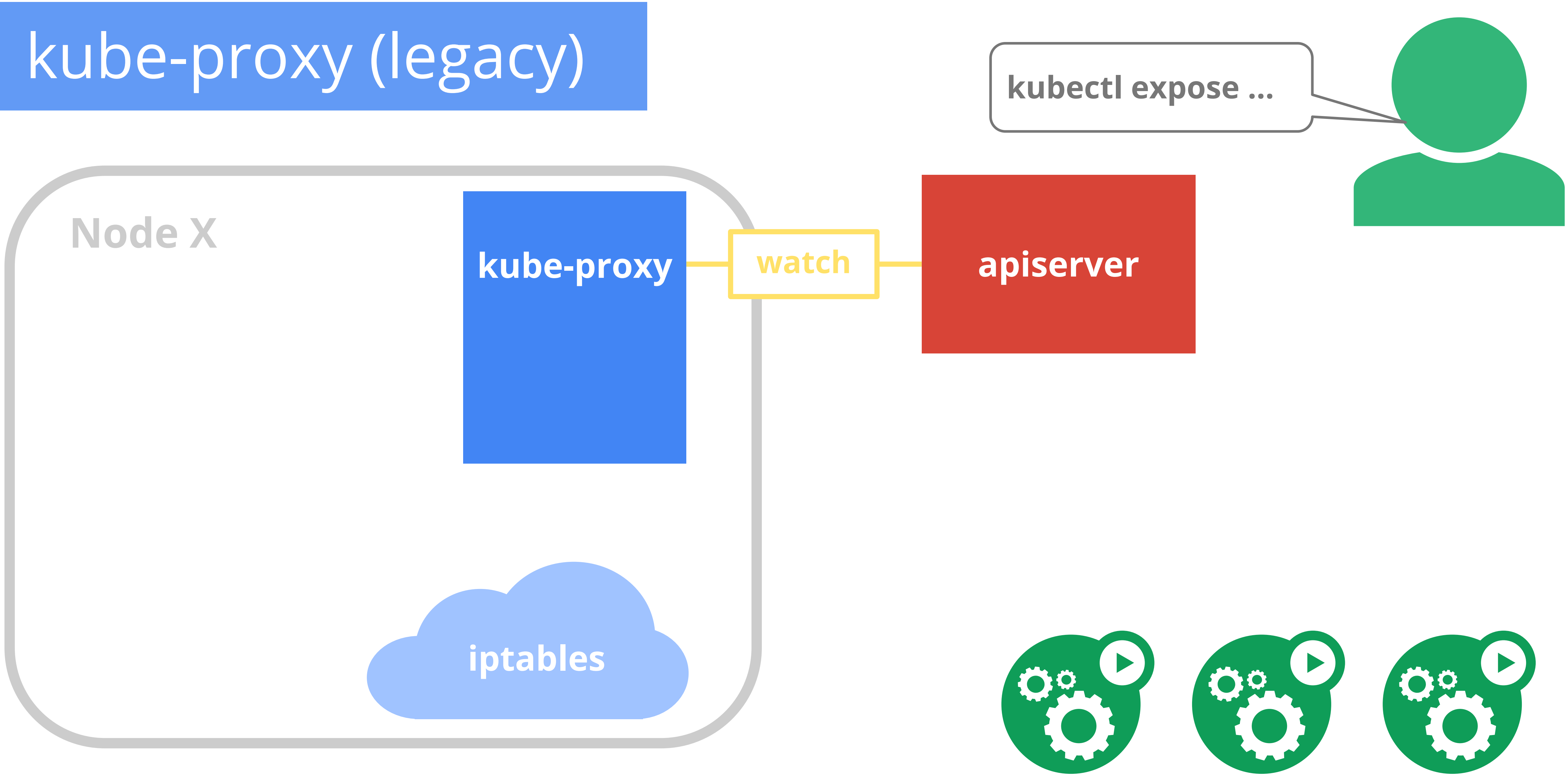
kube-proxy (legacy)



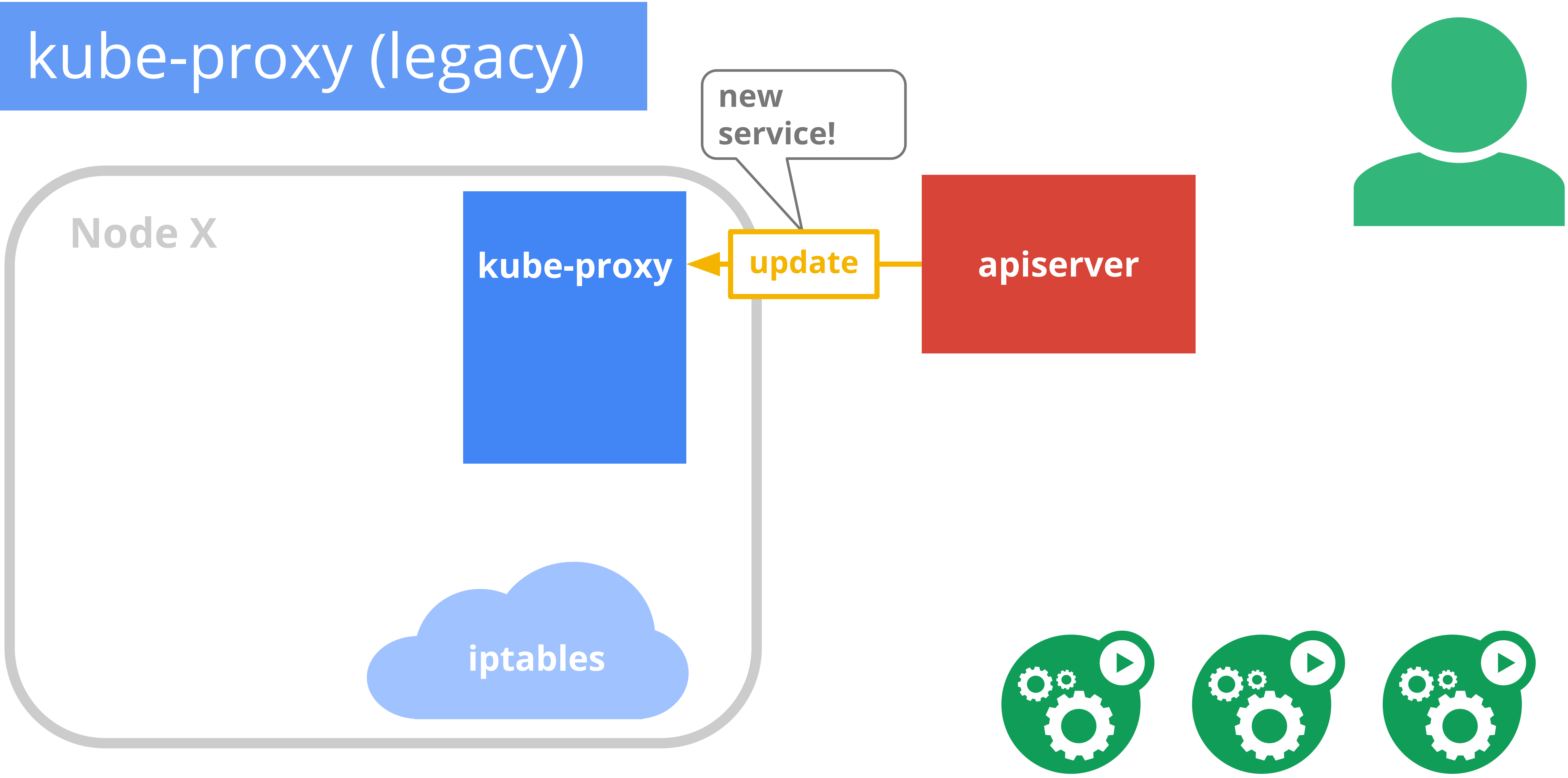
kube-proxy (legacy)



kube-proxy (legacy)



kube-proxy (legacy)


new service!

Node X

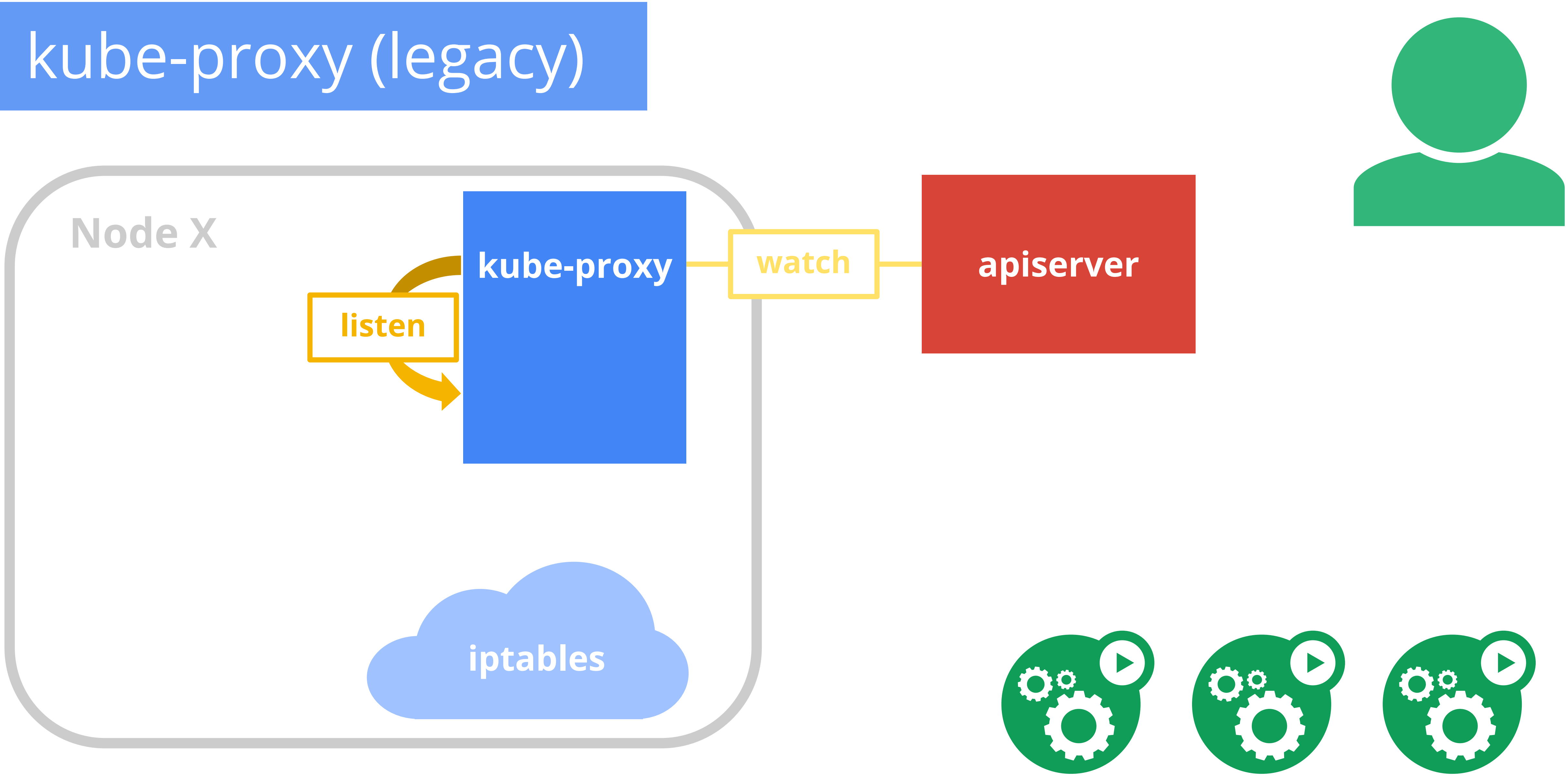
kube-proxy

update

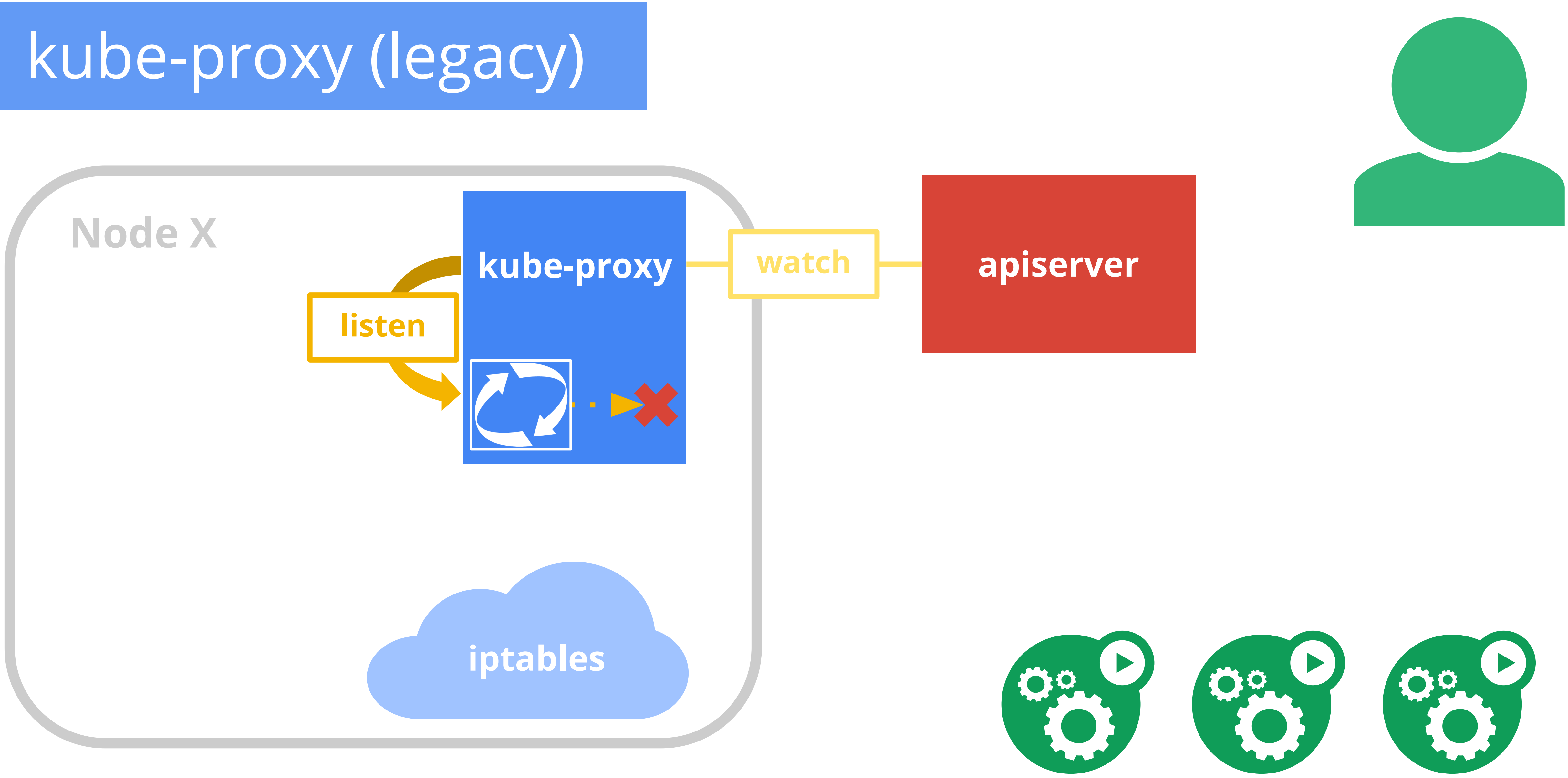
apiserver

iptables

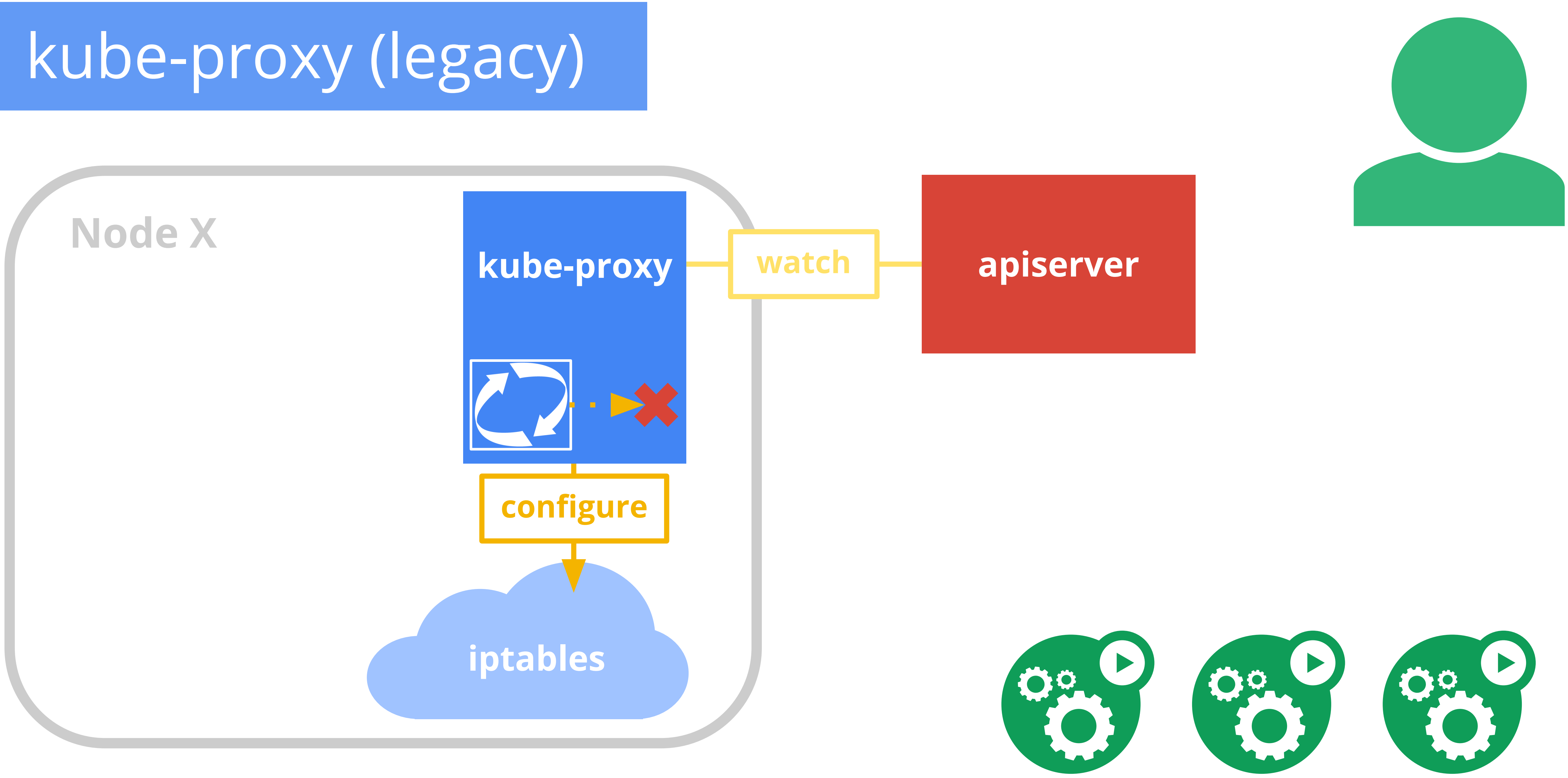
kube-proxy (legacy)



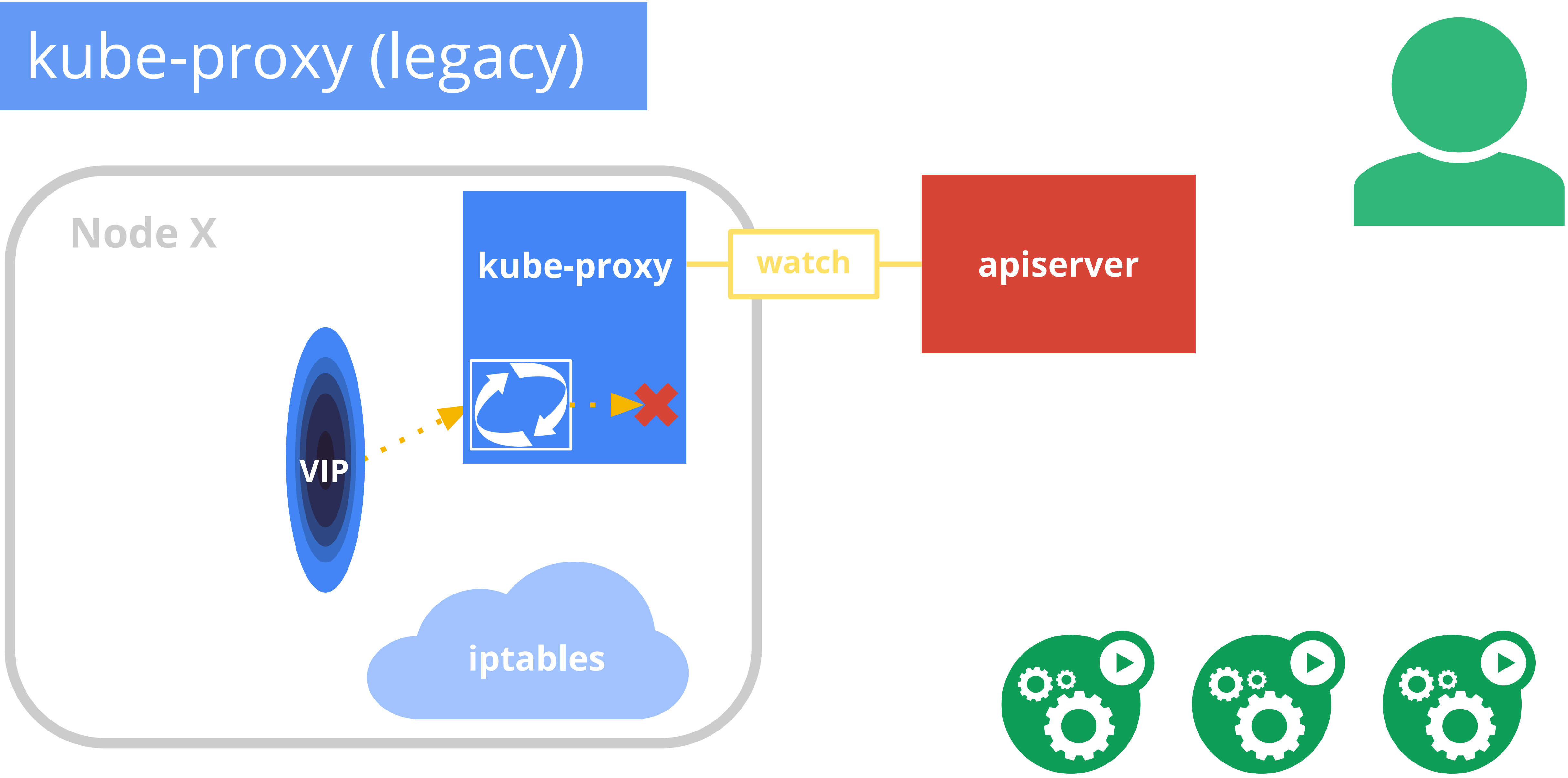
kube-proxy (legacy)



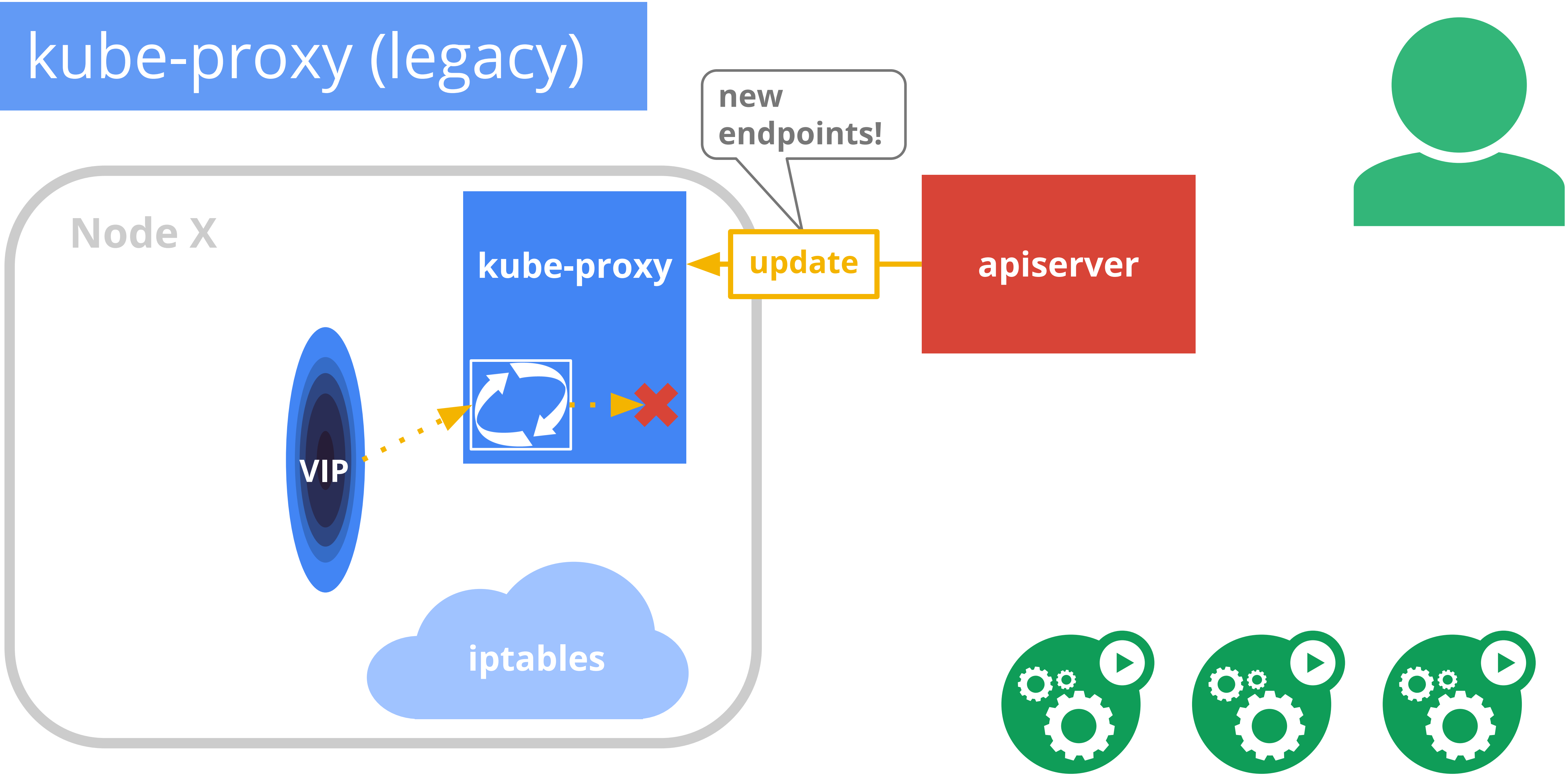
kube-proxy (legacy)



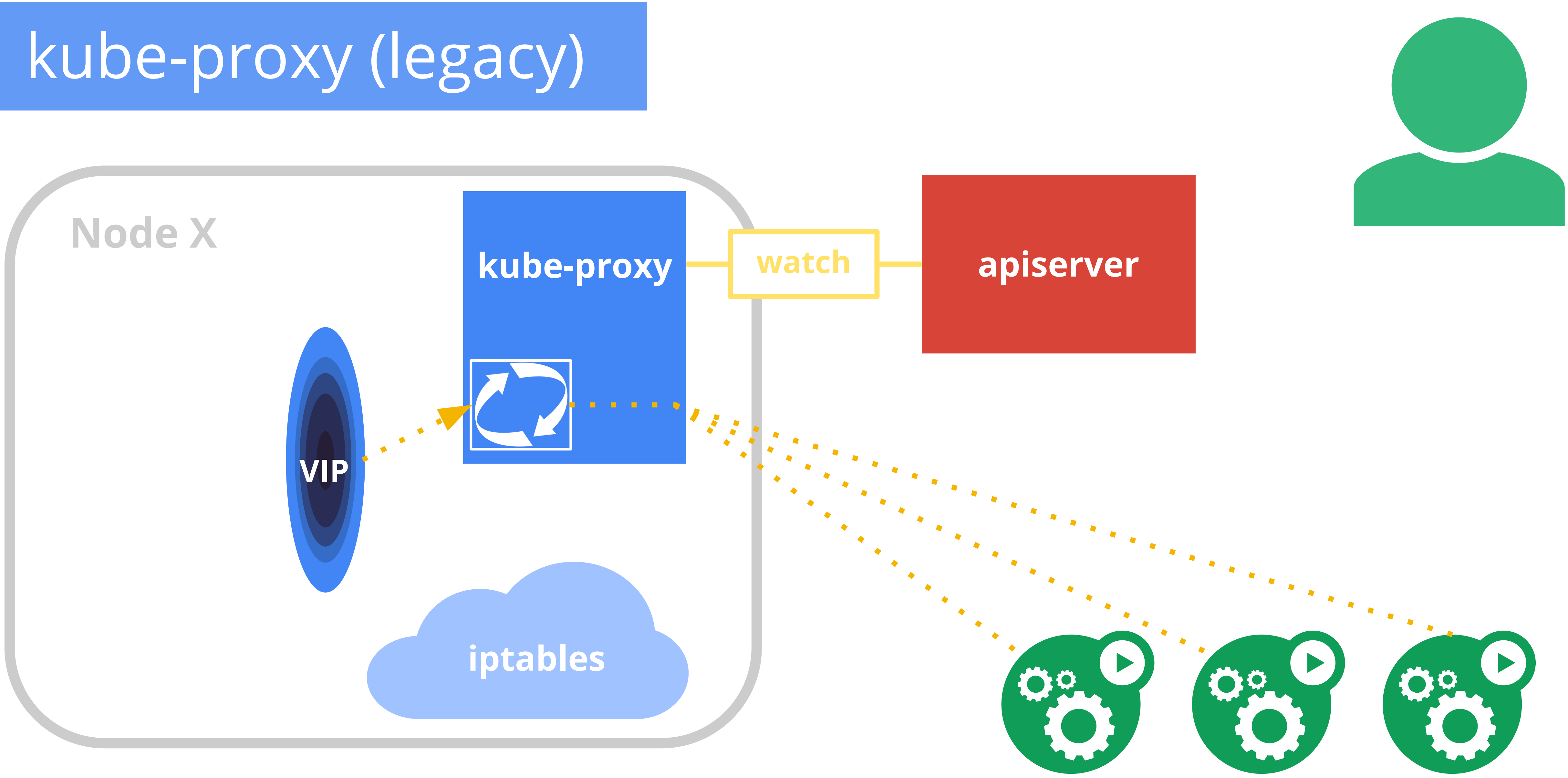
kube-proxy (legacy)



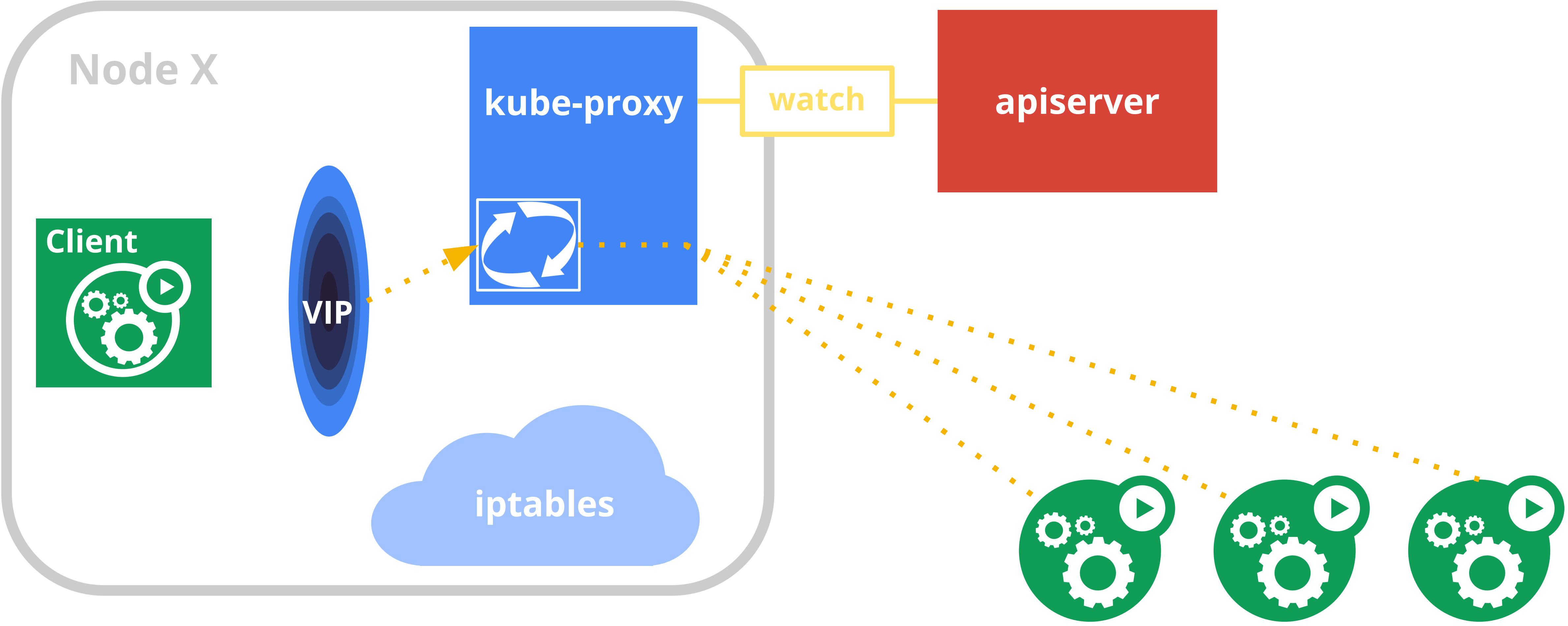
kube-proxy (legacy)



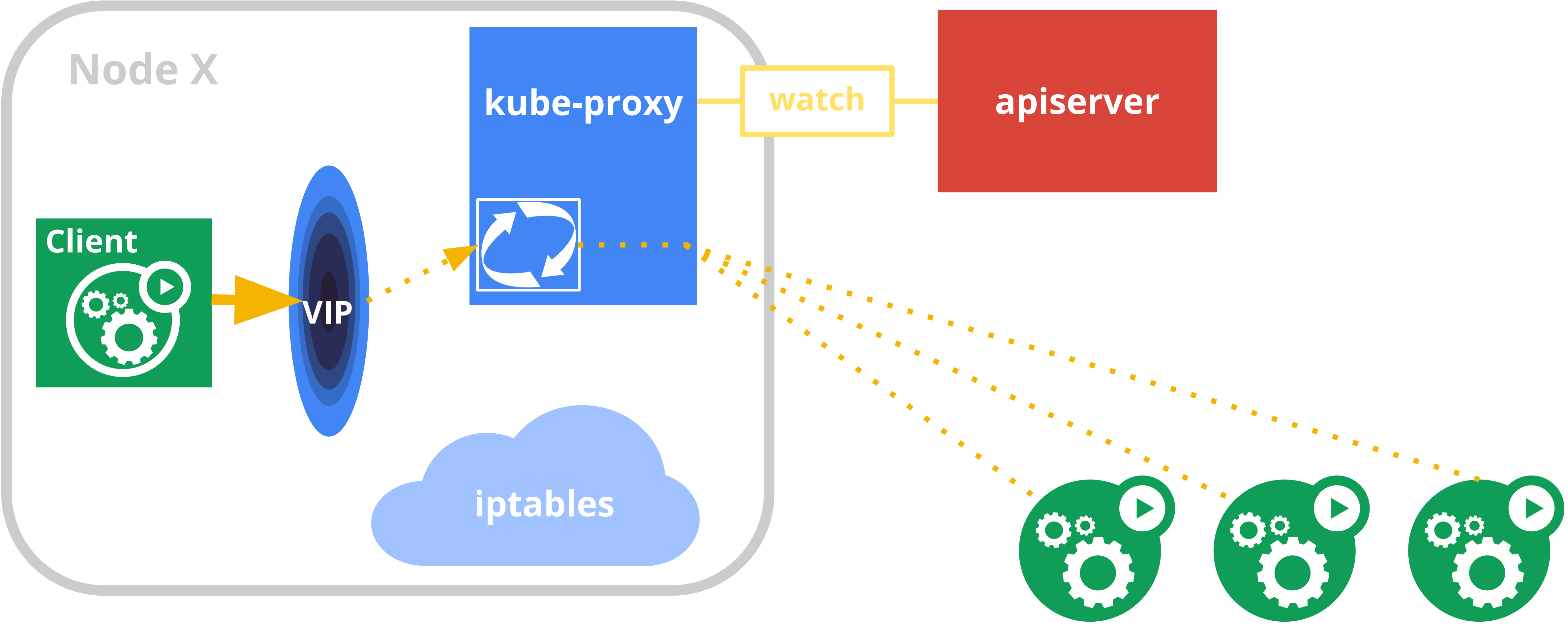
kube-proxy (legacy)



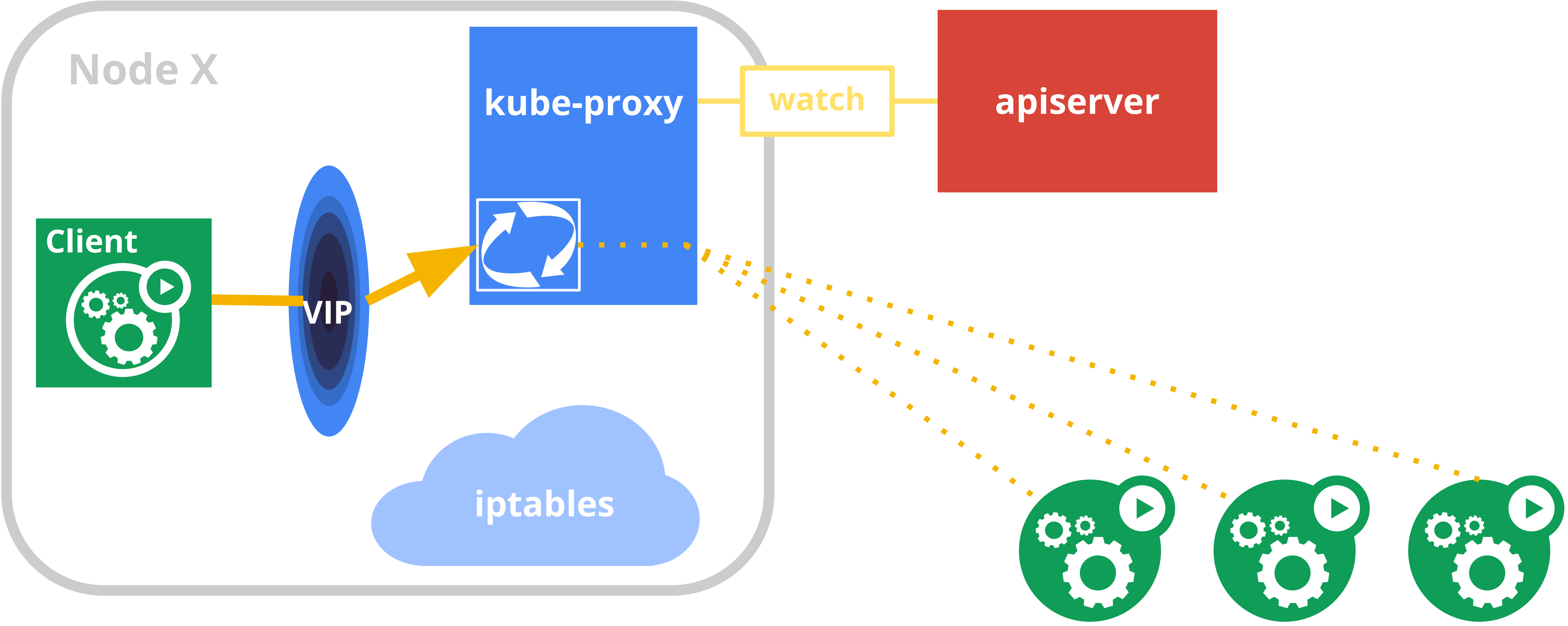
kube-proxy (legacy)



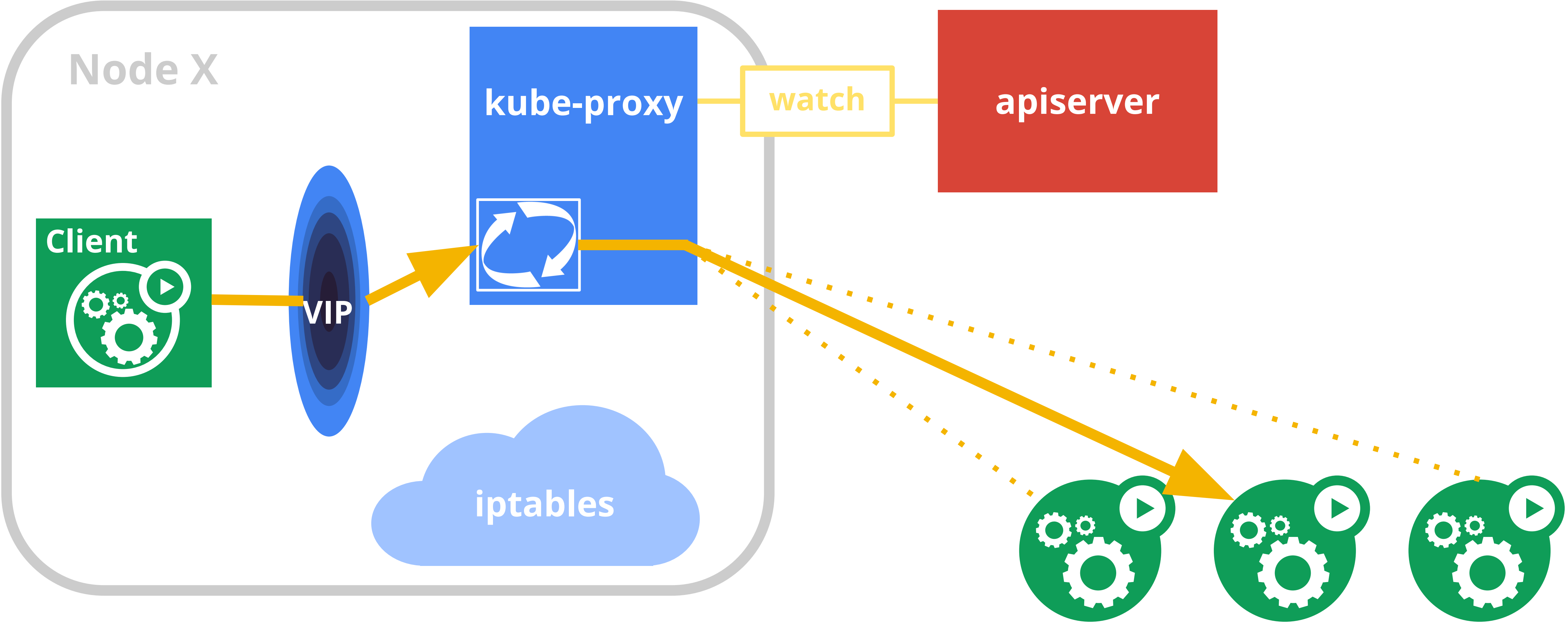
kube-proxy (legacy)



kube-proxy (legacy)



kube-proxy (legacy)



kube-proxy (legacy)

Userspace proxy isn't ideal

Burns CPU copying bytes

- "Proxy" is just parallel copy loops.

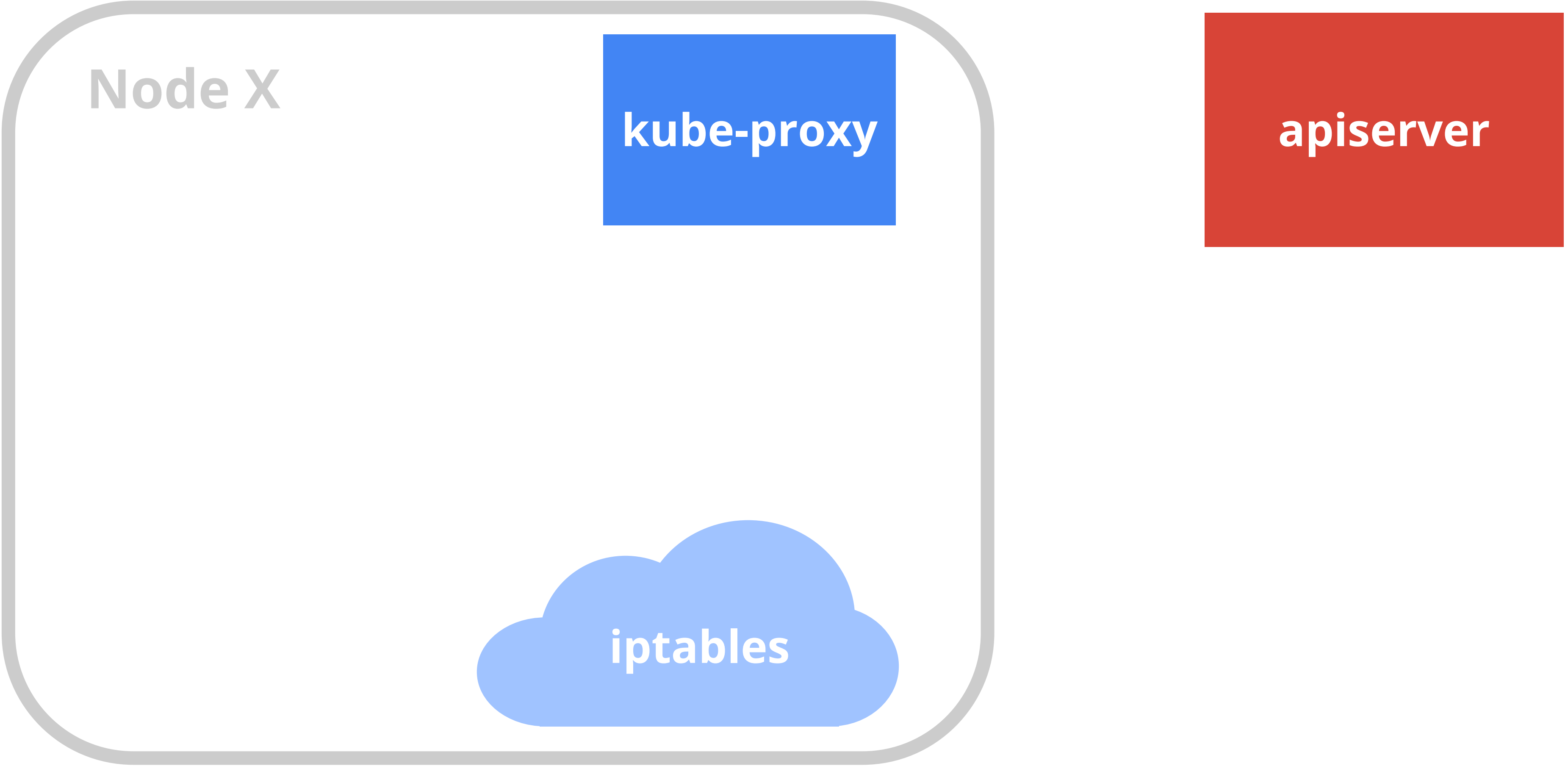
Loses source IP

- Everything looks like it's from the node IP.

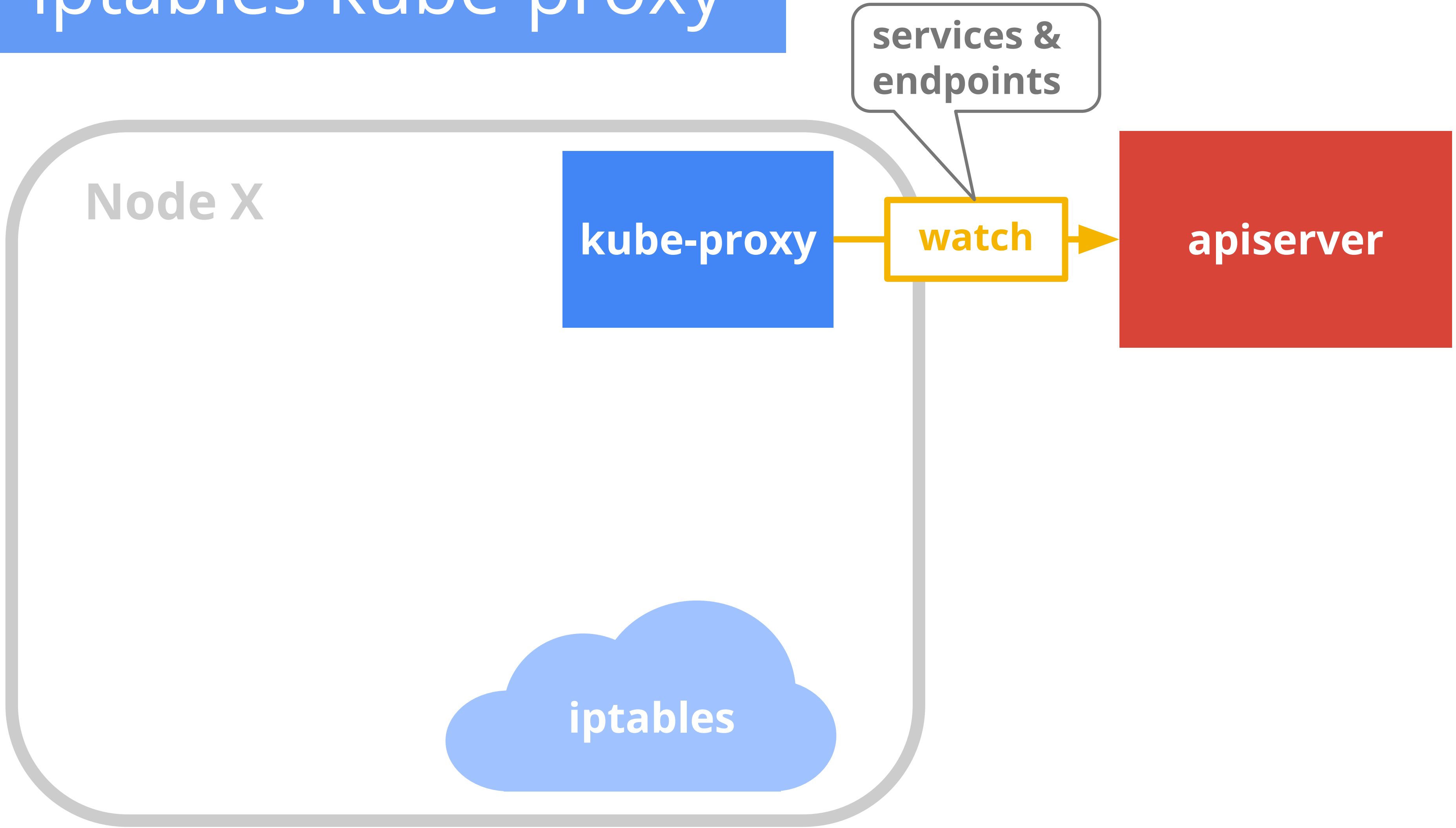
Userspace TCP listening = higher latency

iptables kube-proxy

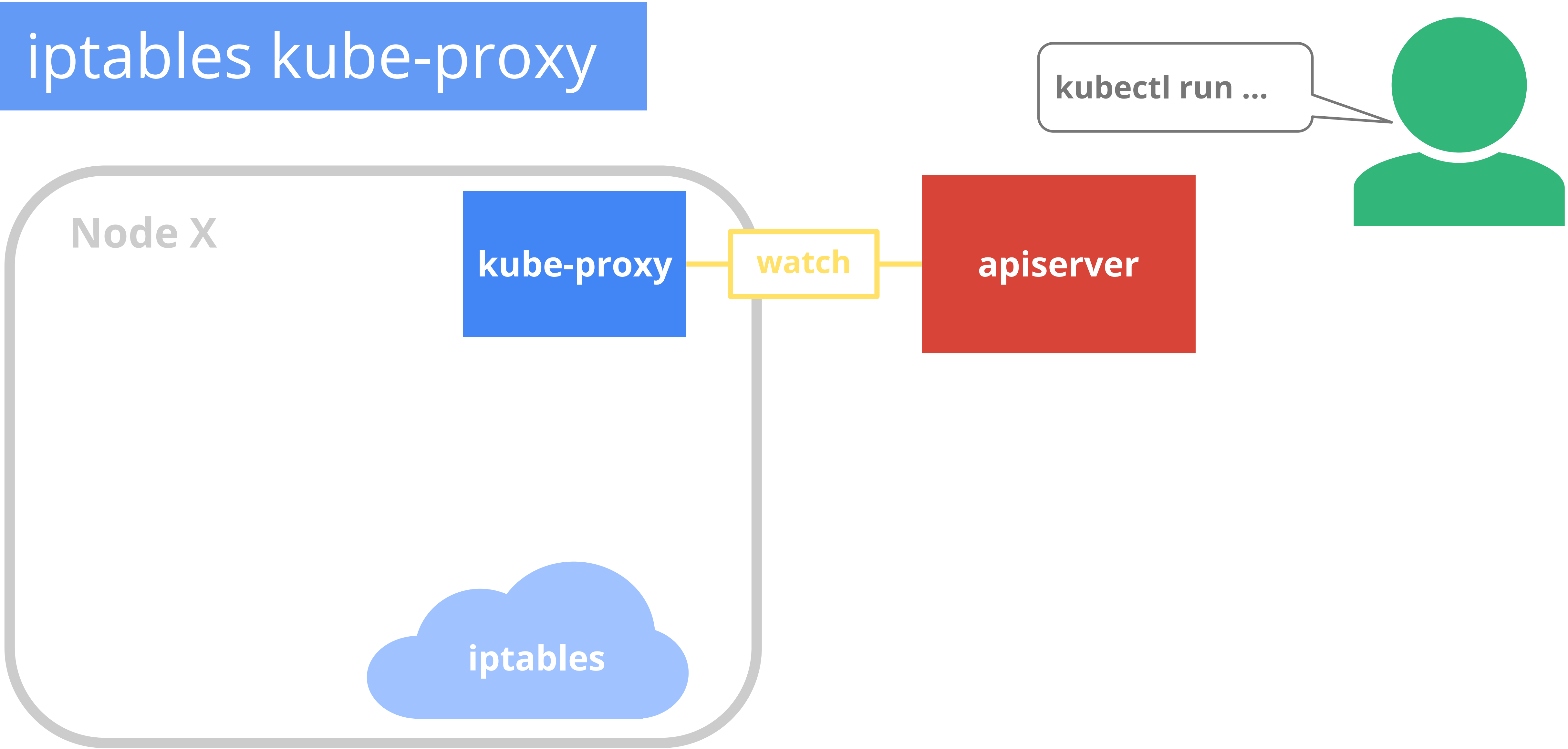
iptables kube-proxy



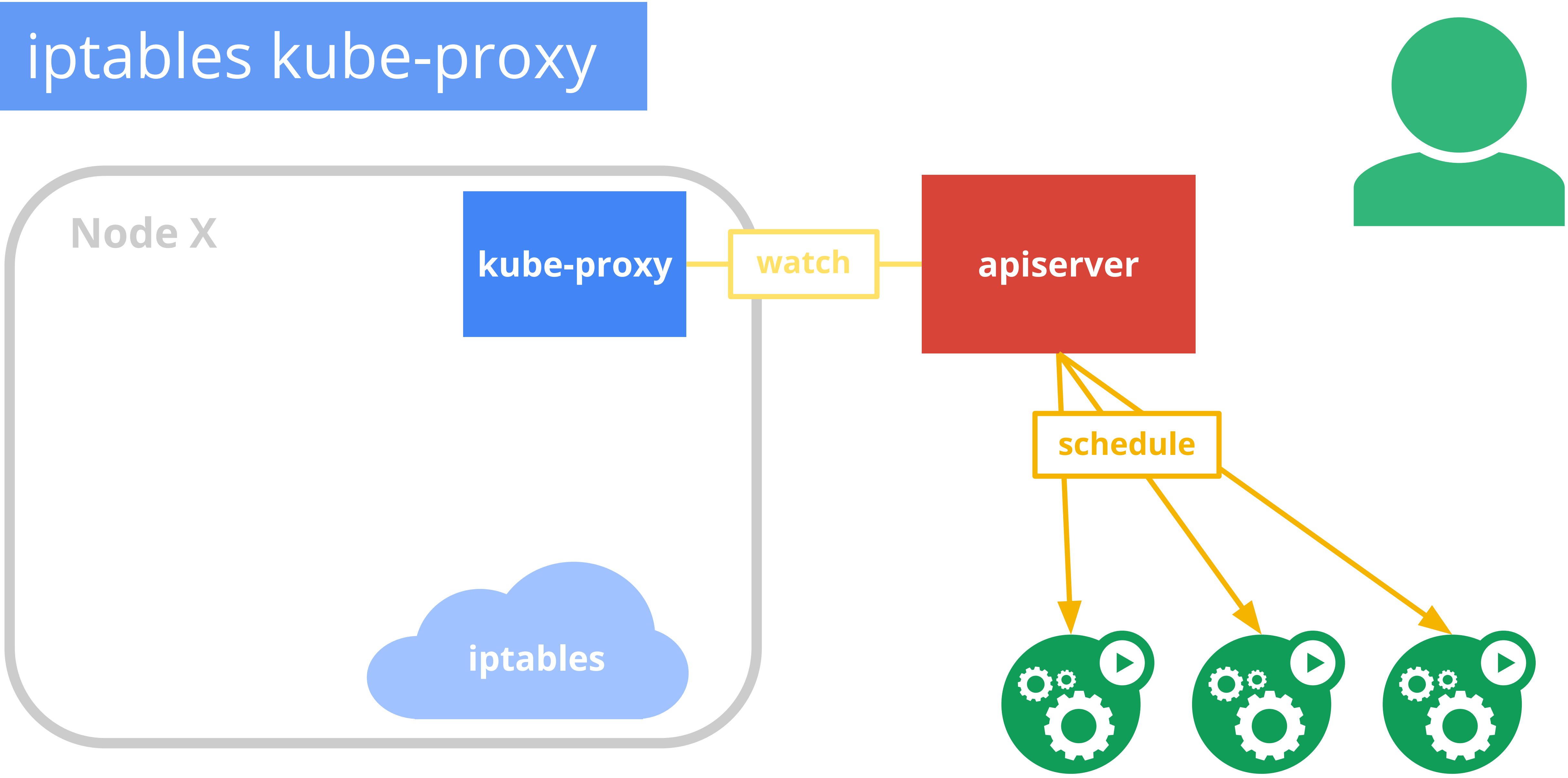
iptables kube-proxy



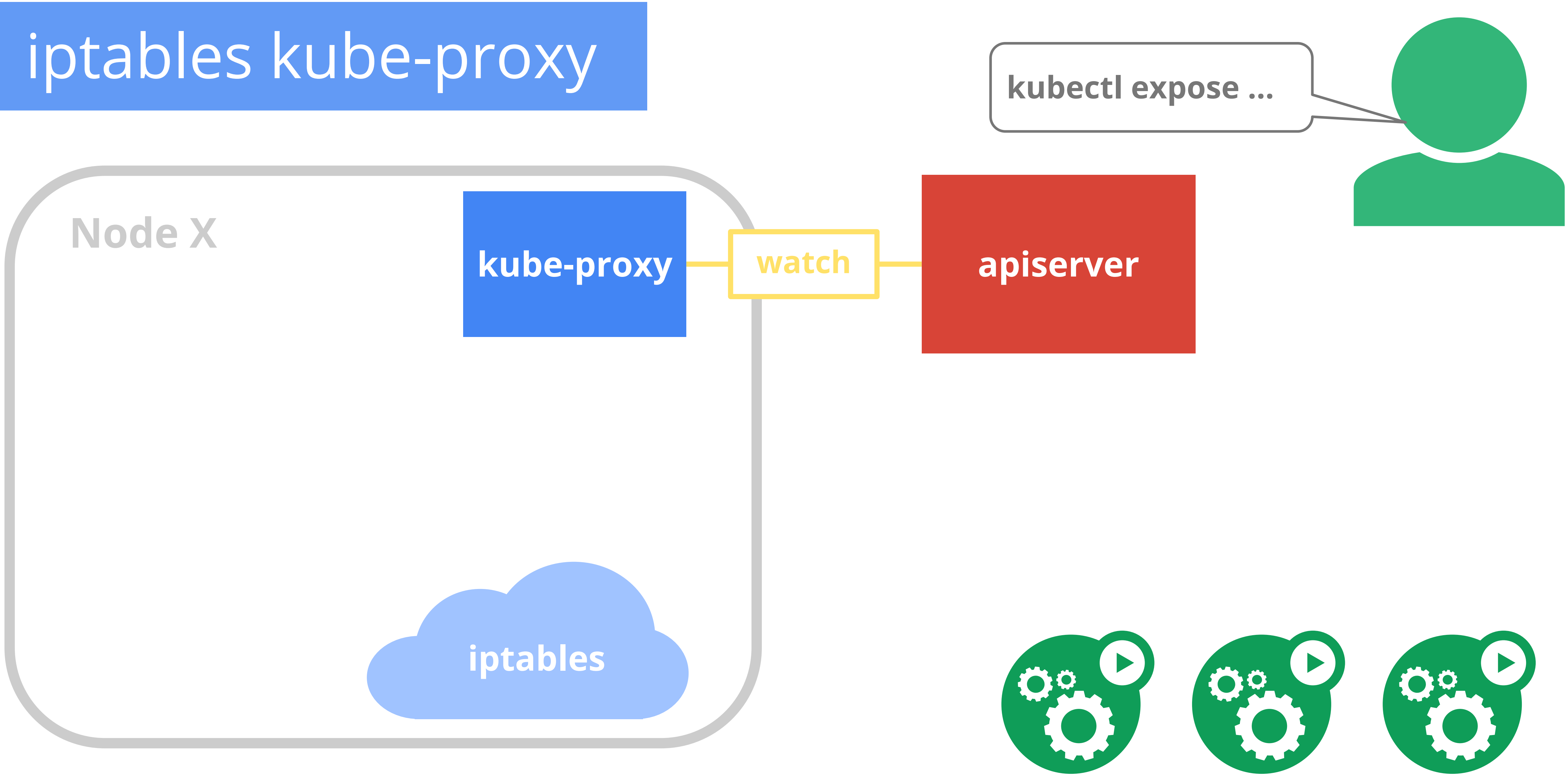
iptables kube-proxy



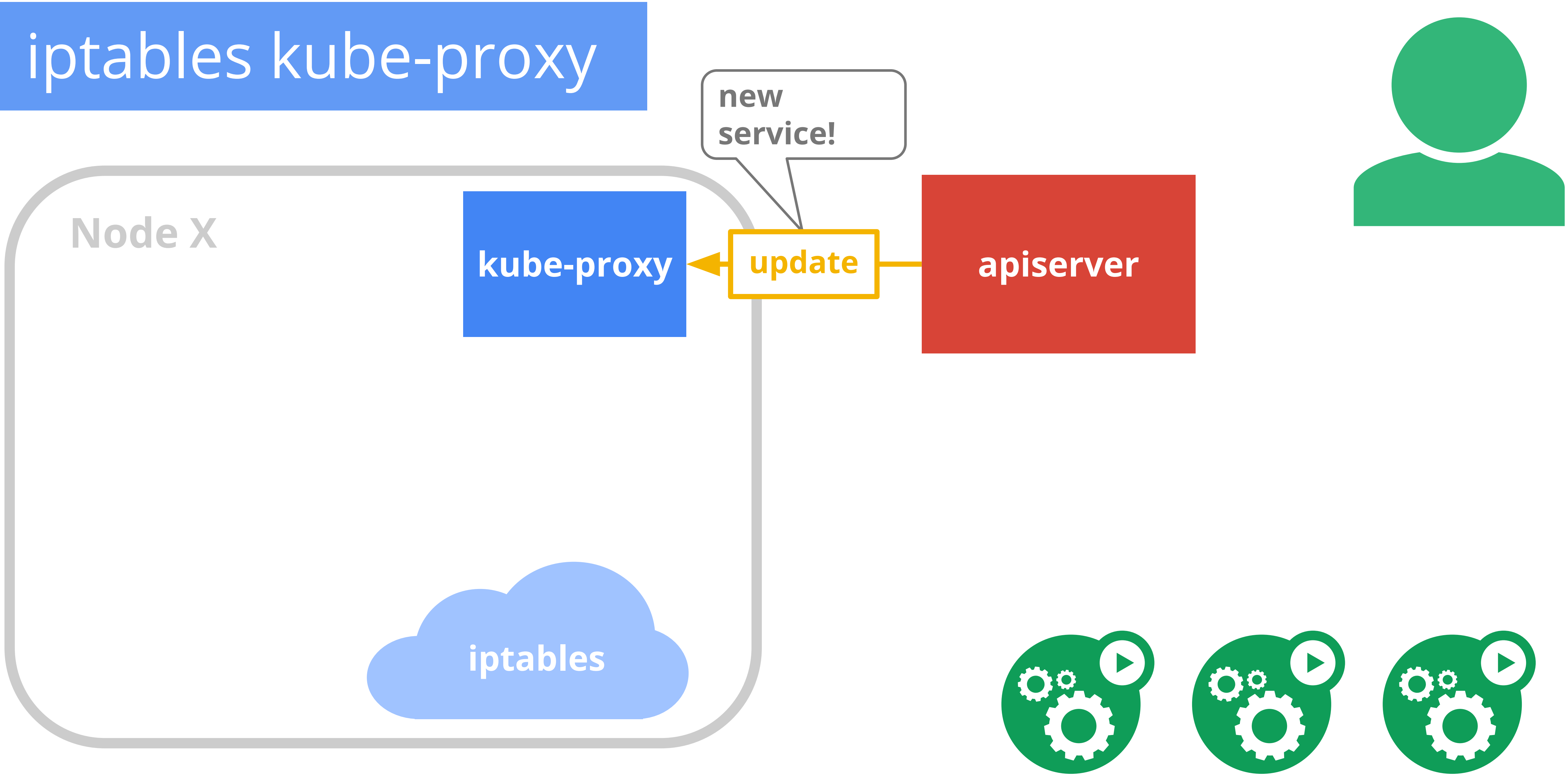
iptables kube-proxy



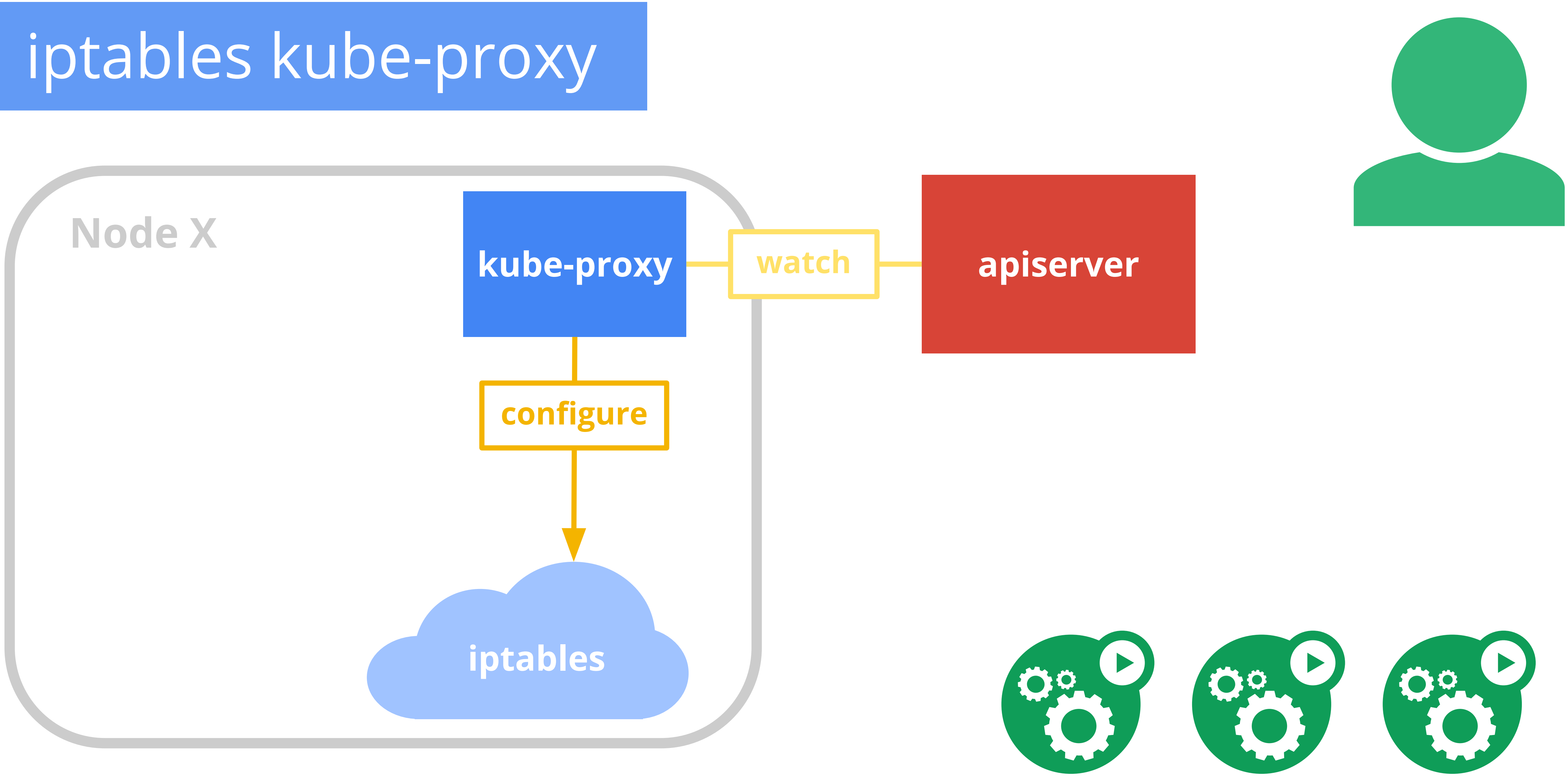
iptables kube-proxy



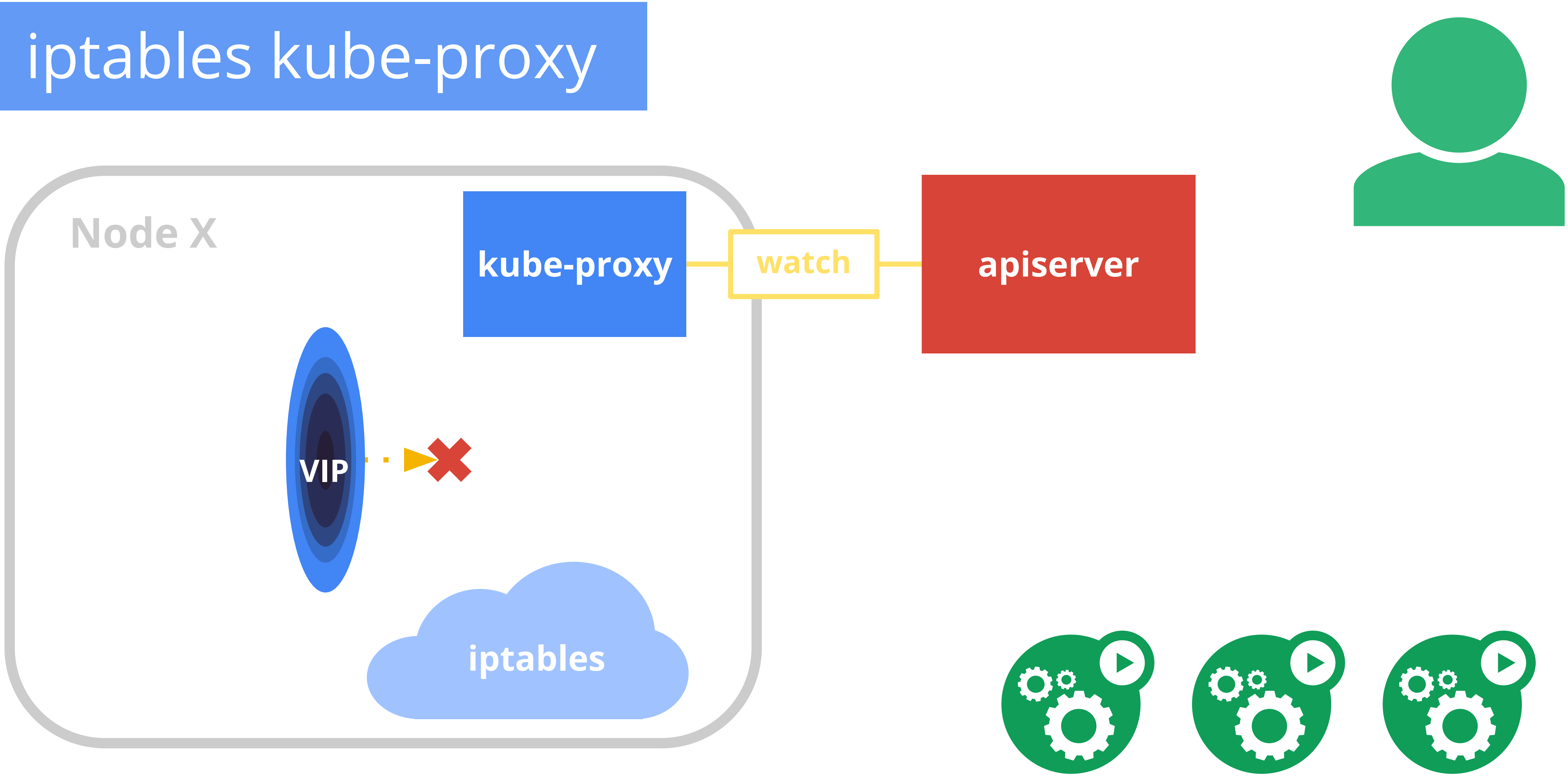
iptables kube-proxy



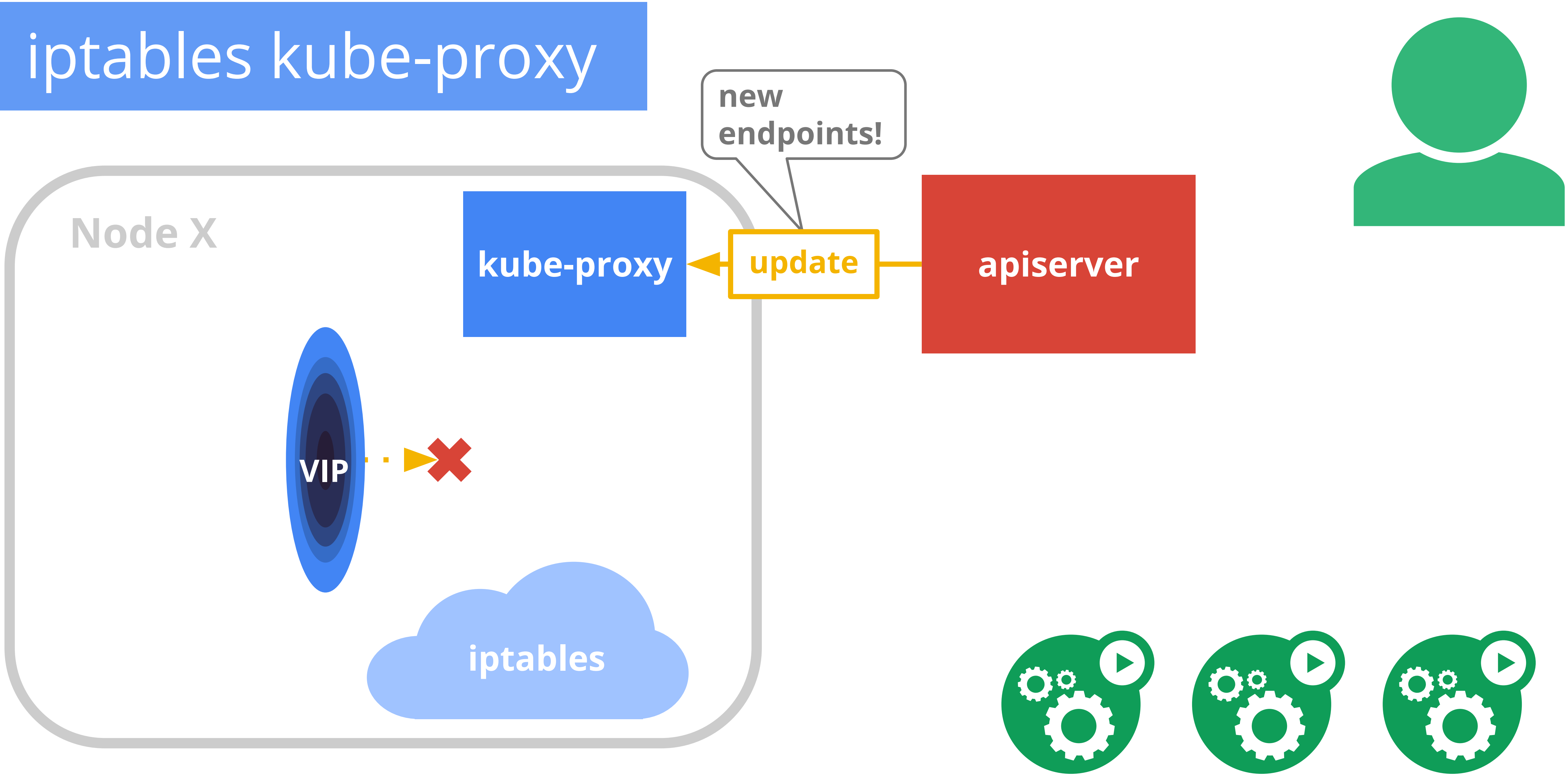
iptables kube-proxy



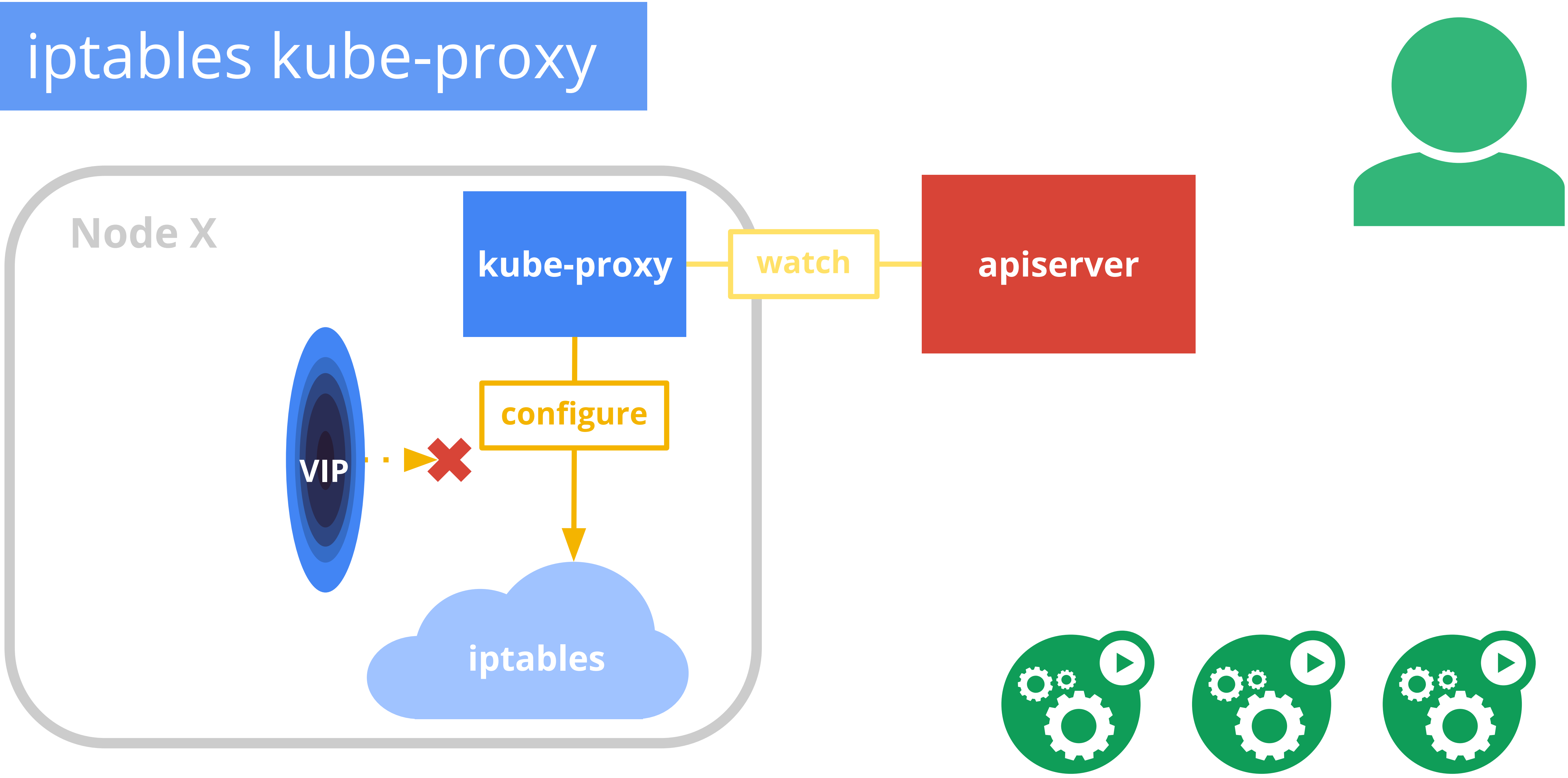
iptables kube-proxy



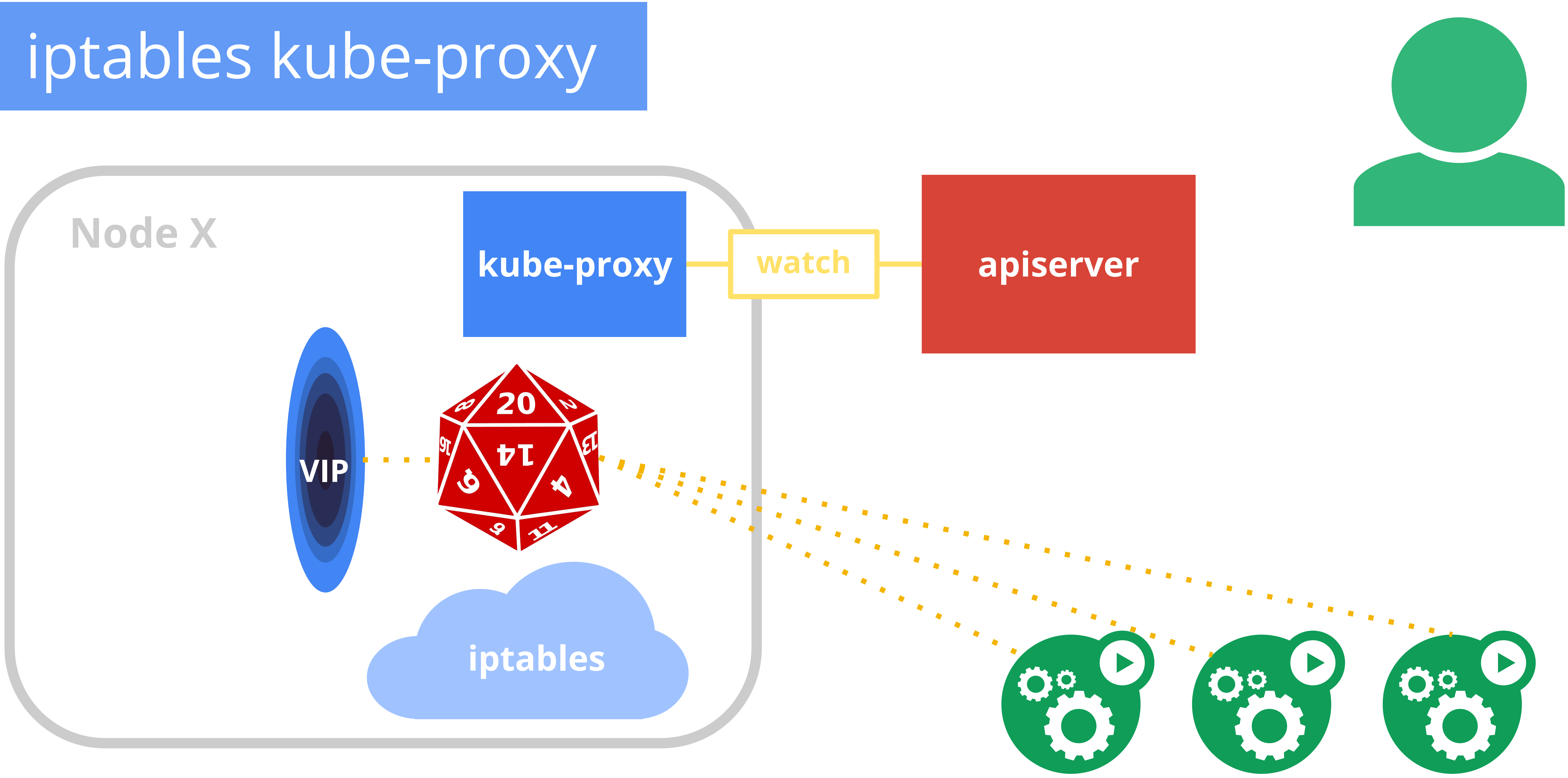
iptables kube-proxy



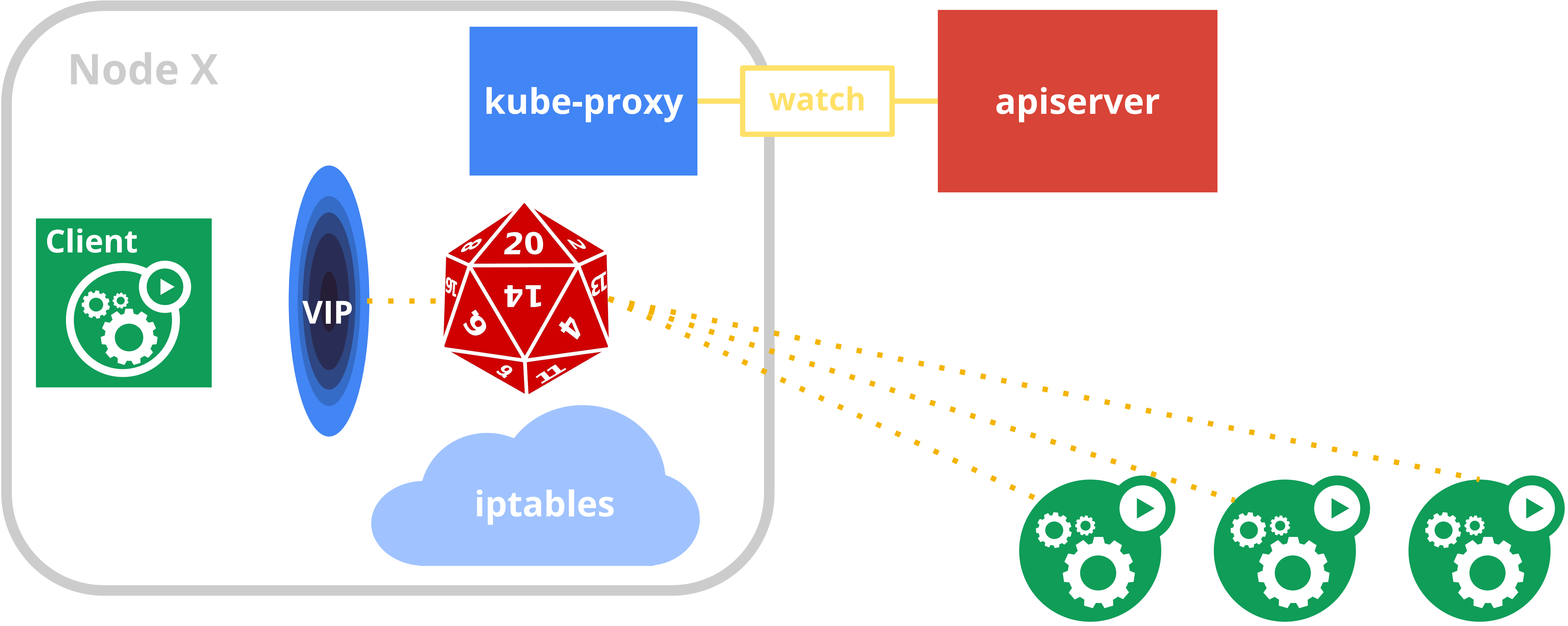
iptables kube-proxy



iptables kube-proxy



iptables kube-proxy



Node X

kube-proxy

watch

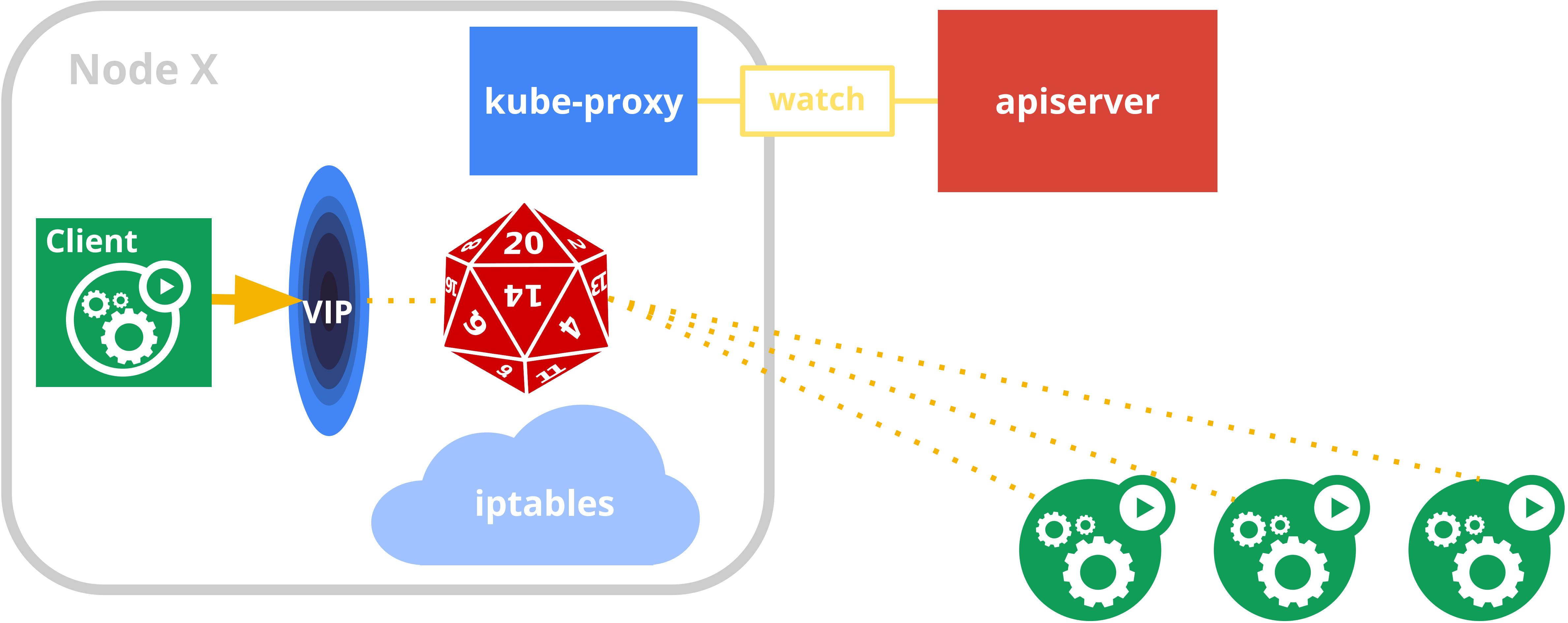
apiserver

Client

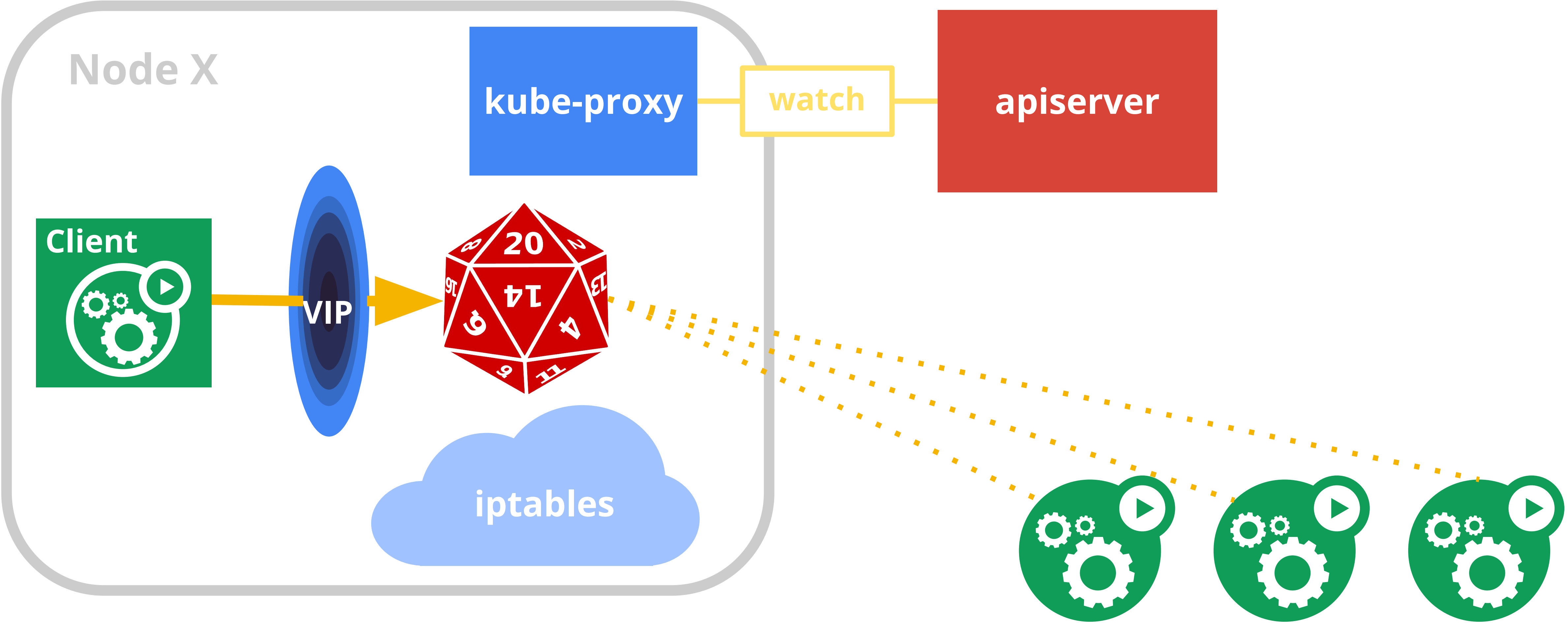
VIP

iptables

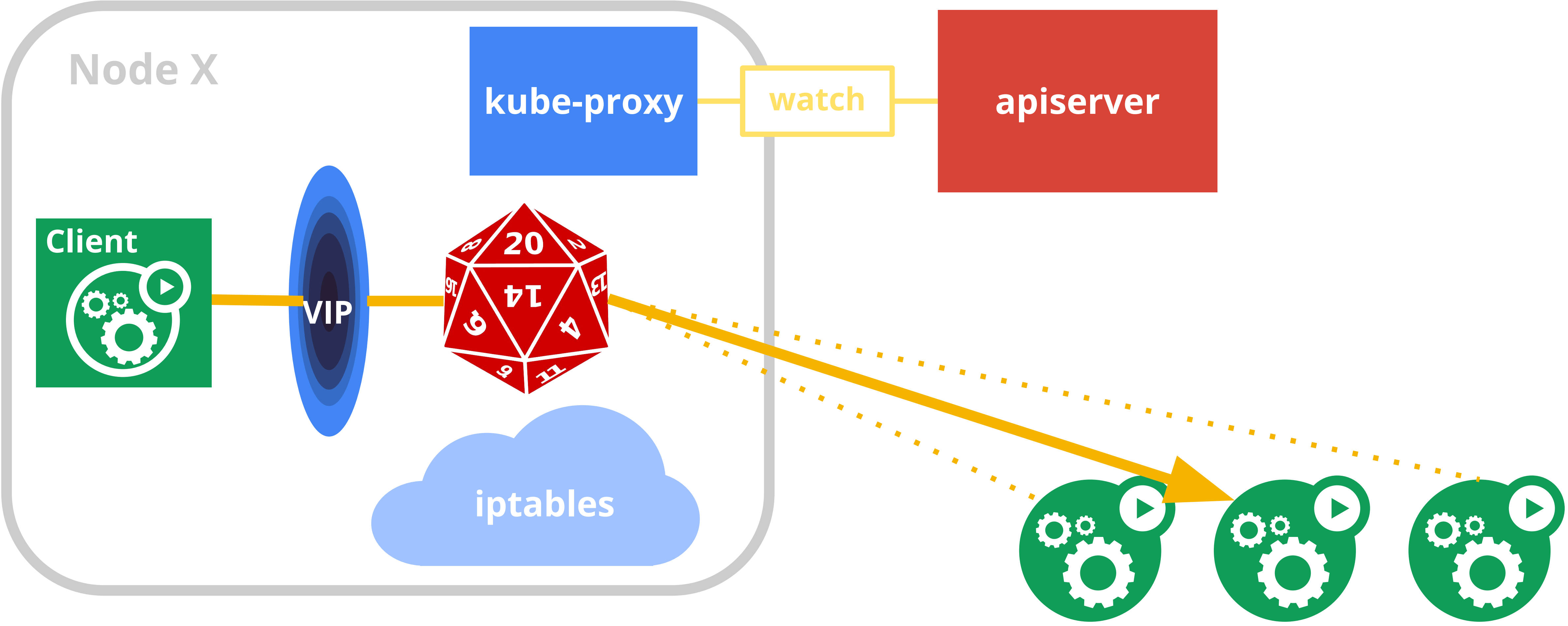
iptables kube-proxy



iptables kube-proxy



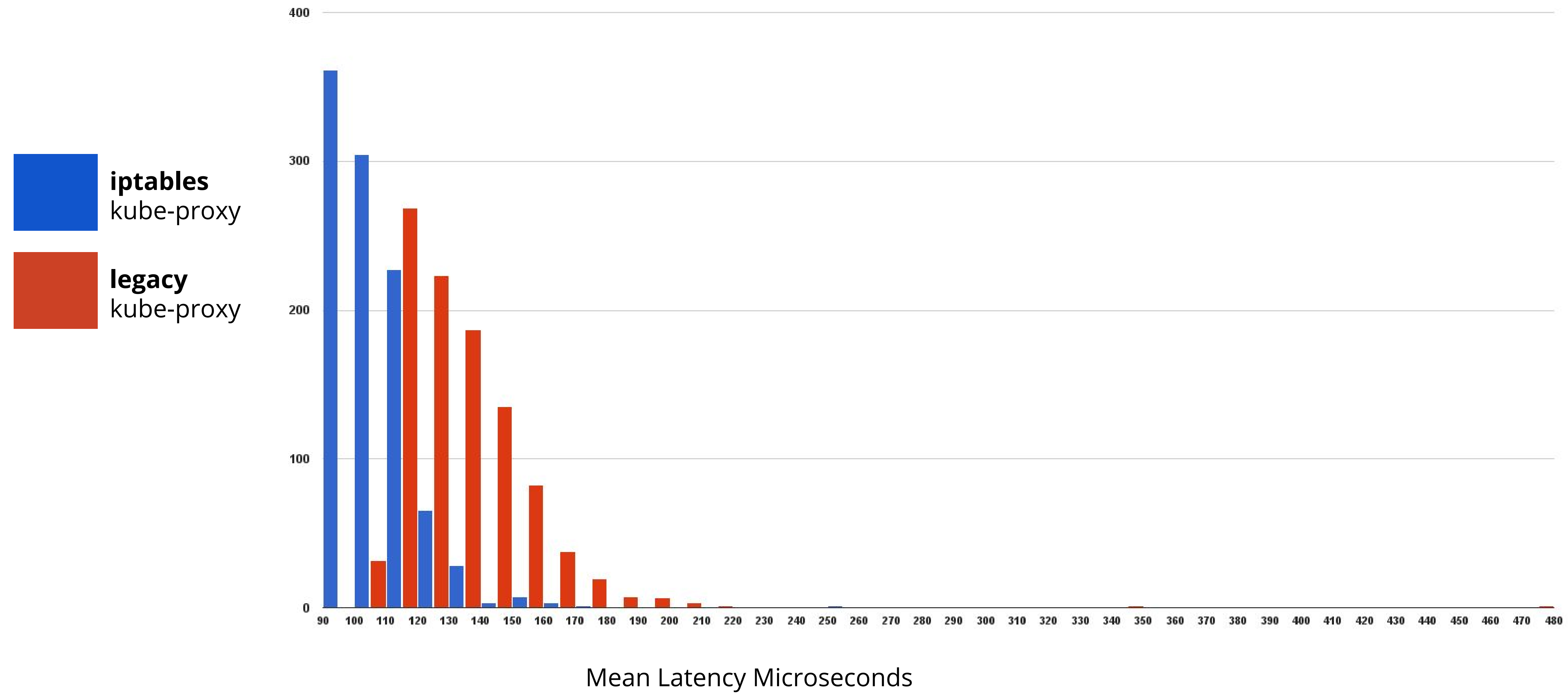
iptables kube-proxy



iptables kube-proxy

Mean Latency

`contrib/for-tests/netperf-tester --number=1000`



Services

Services are just an abstraction

- Only requirement: route (and maybe load balance) a virtual IP to a set of backends.

Kube-proxy is an implementation

- Kube-proxy watches apiserver.
- iptables is re-configured on changes.

There could be other ways

- Userspace, iptables, IP Virtual Servers?

DNS

Run SkyDNS as a pod in the cluster

- kube2sky bridges Kubernetes API -> SkyDNS
- Tell kubelets about it (static service IP)

Strictly optional, but practically required

- LOTS of things depend on it
- Probably will become more integrated

Or plug in your own!

```
kubernetes

kubernetes.default

kubernetes.default.svc.cluster.local

foo.my-namespace.svc.cluster.local
```

DNS

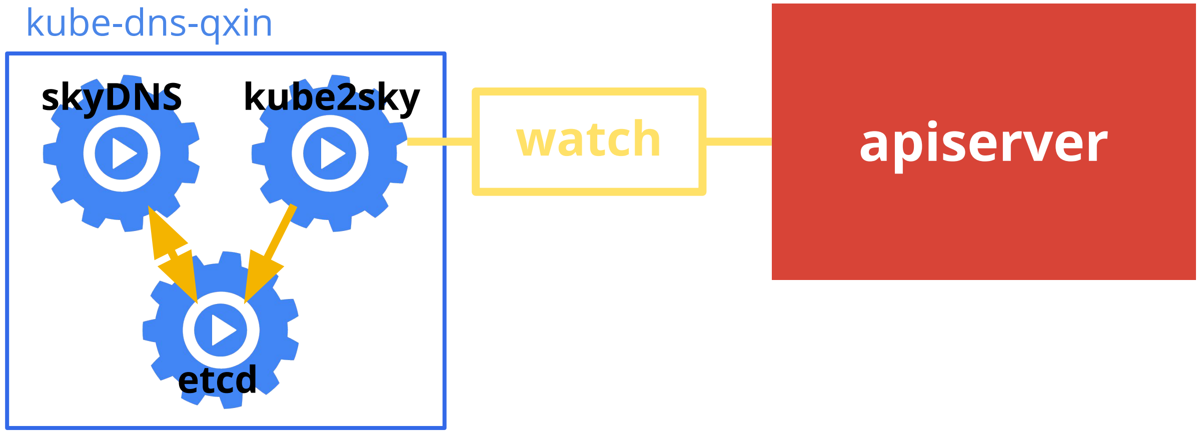
Run SkyDNS as a pod in the cluster

- kube2sky bridges Kubernetes API -> SkyDNS
- Tell kubelets about it (static service IP)

Strictly optional, but practically required

- LOTS of things depend on it
- Probably will become more integrated

Or plug in your own!



DNS

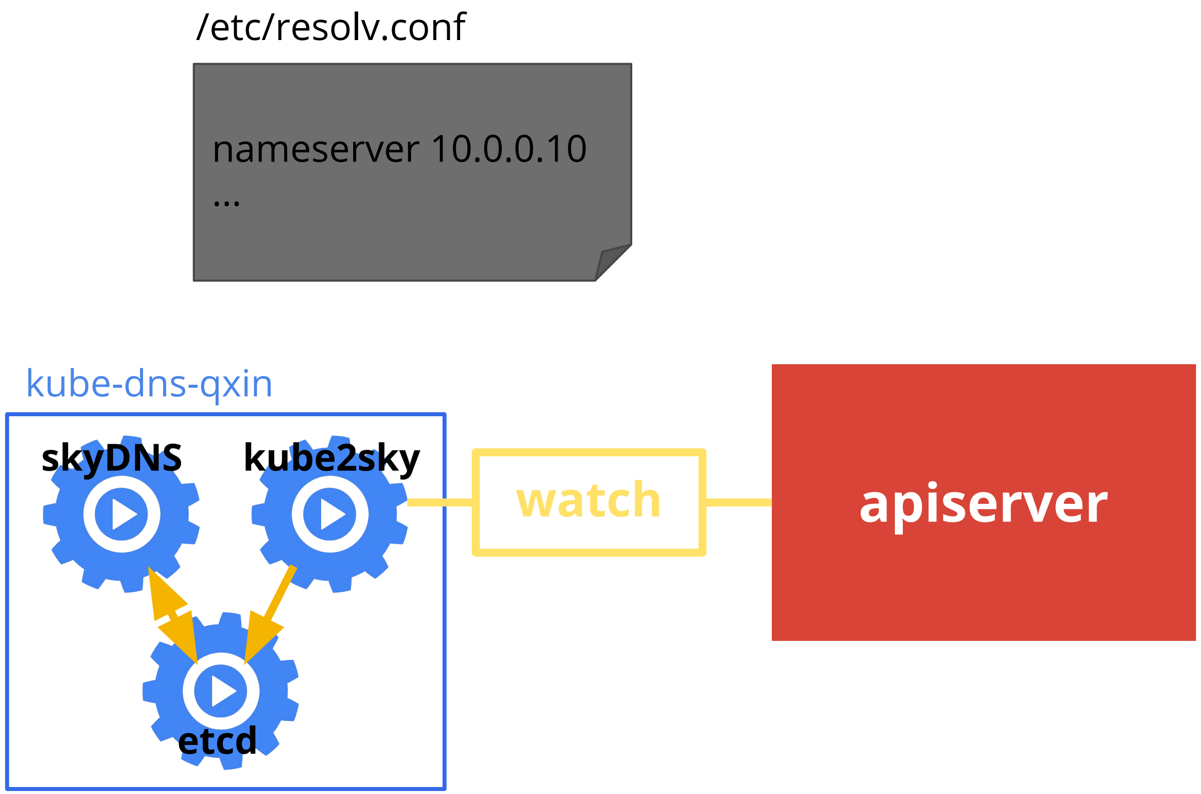
Run SkyDNS as a pod in the cluster

- kube2sky bridges Kubernetes API -> SkyDNS
- Tell kubelets about it (static service IP)

Strictly optional, but practically required

- LOTS of things depend on it
- Probably will become more integrated

Or plug in your own!



/etc/resolv.conf

```
nameserver 10.0.0.10
…
```

DNS

Run SkyDNS as a pod in the cluster

- kube2sky bridges Kubernetes API -> SkyDNS
- Tell kubelets about it (static service IP)

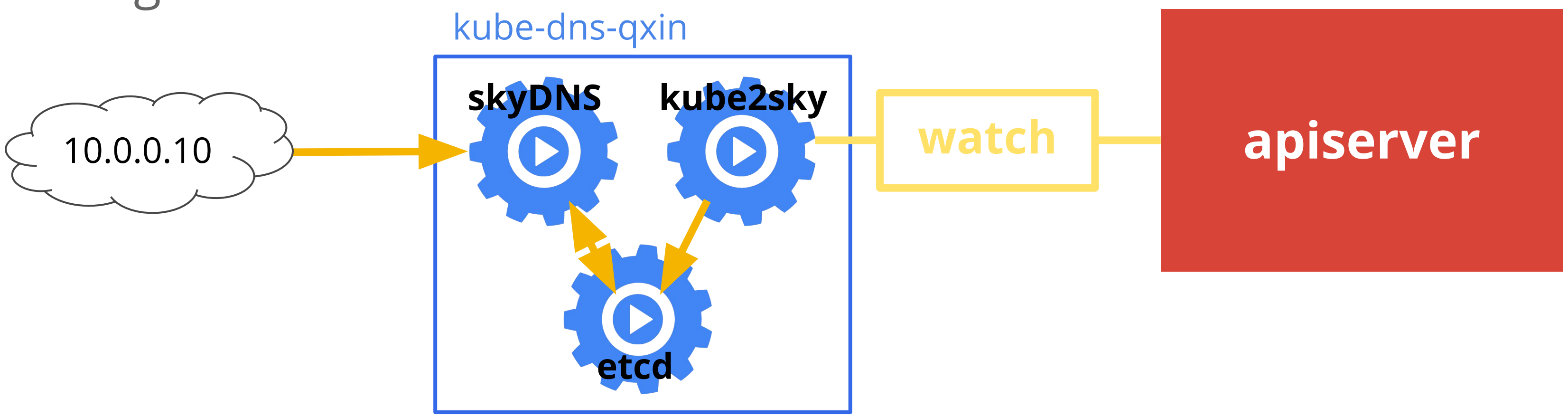
Strictly optional, but practically required

- LOTS of things depend on it
- Probably will become more integrated

Or plug in your own!

/etc/resolv.conf

```
nameserver 10.0.0.10
…
```



Putting it Together

What happens when I...

```
$ curl foo.my-namespace
```

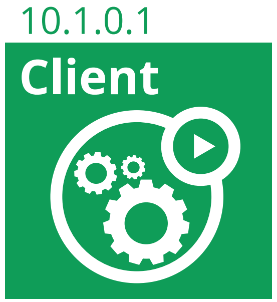
Putting it Together

What happens when I...

```
$ curl foo.my-namespace
```

/etc/resolv.conf

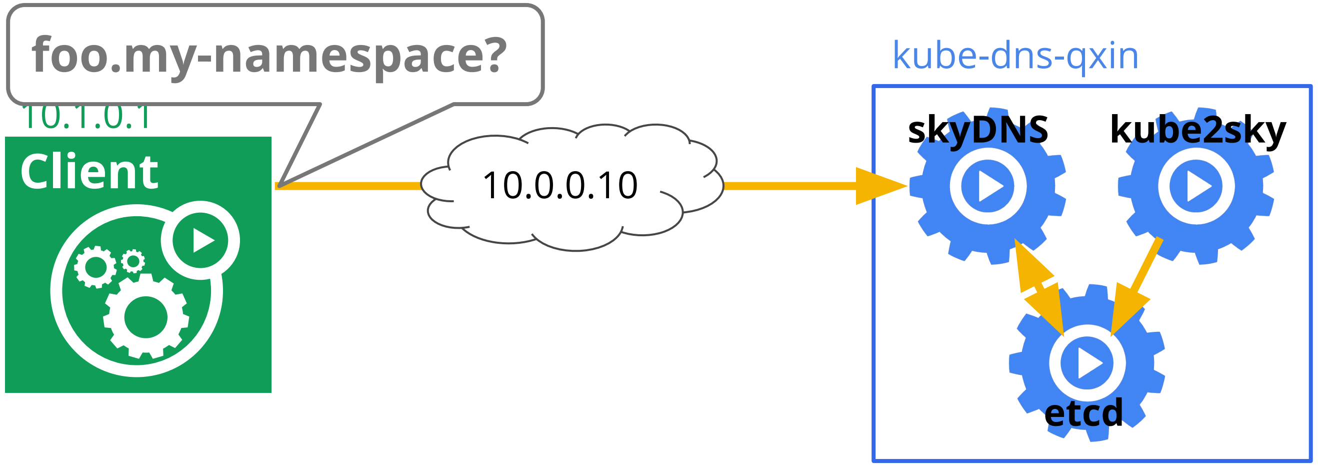
```
nameserver 10.0.0.10
...
```



Putting it Together

What happens when I...

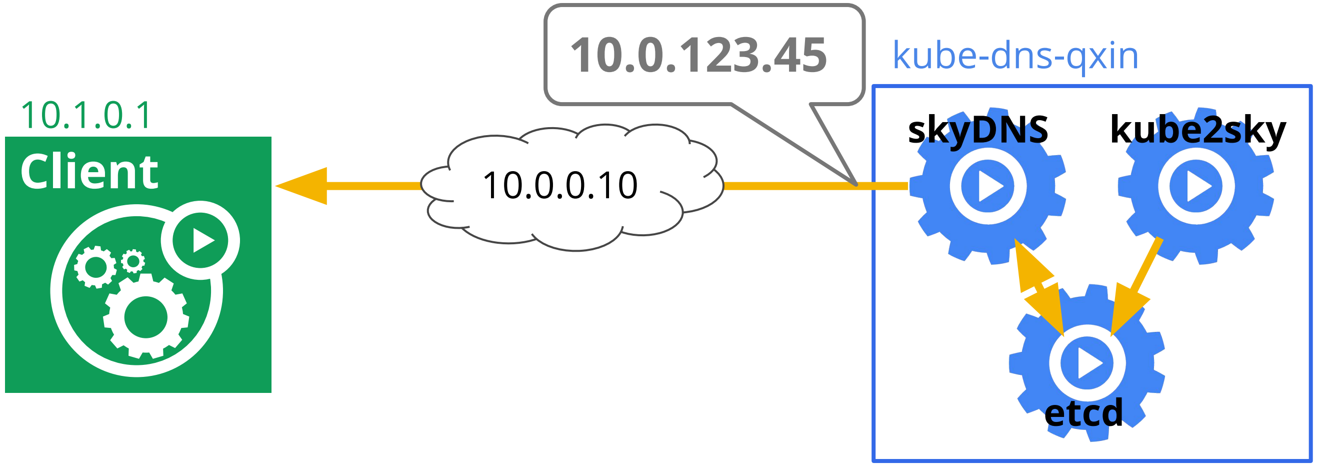
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

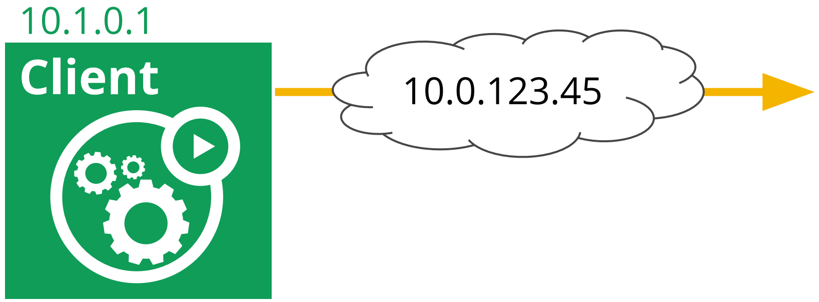
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

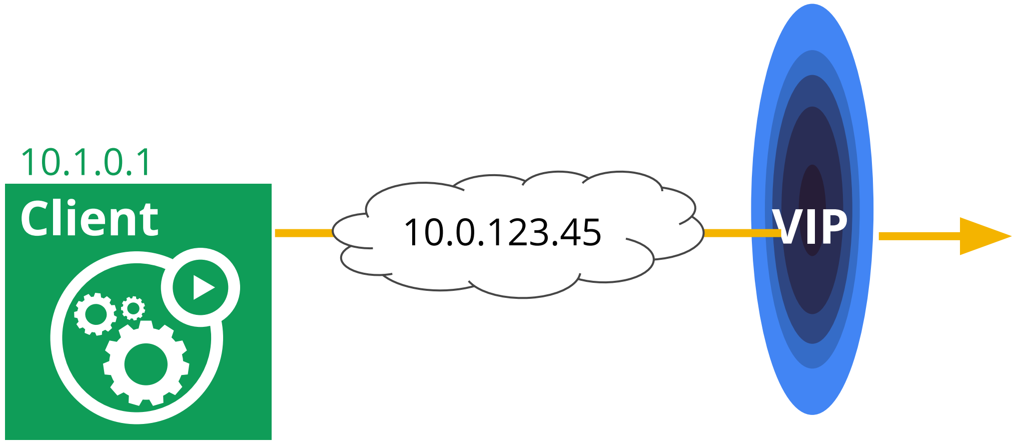
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

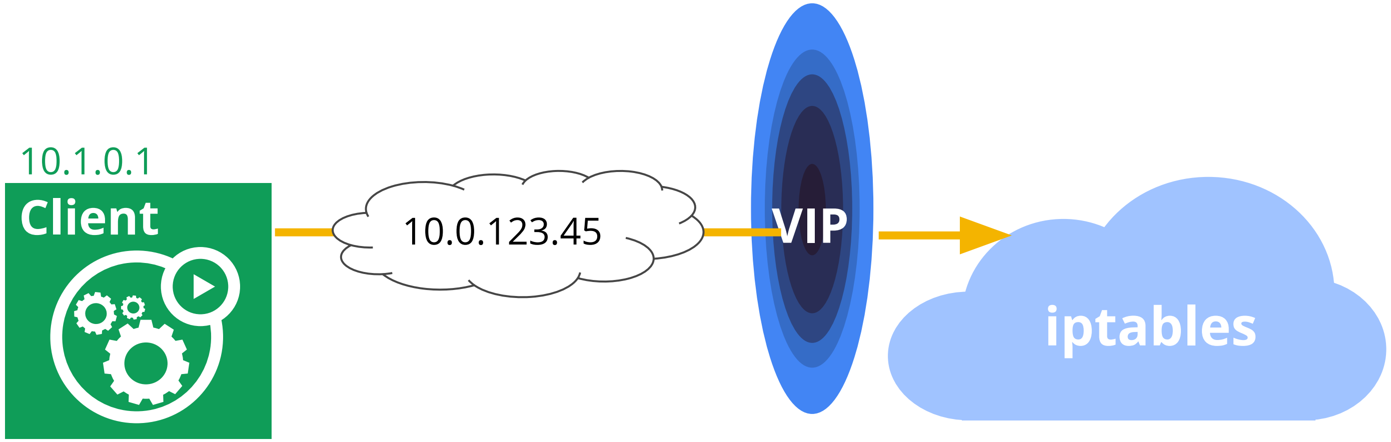
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

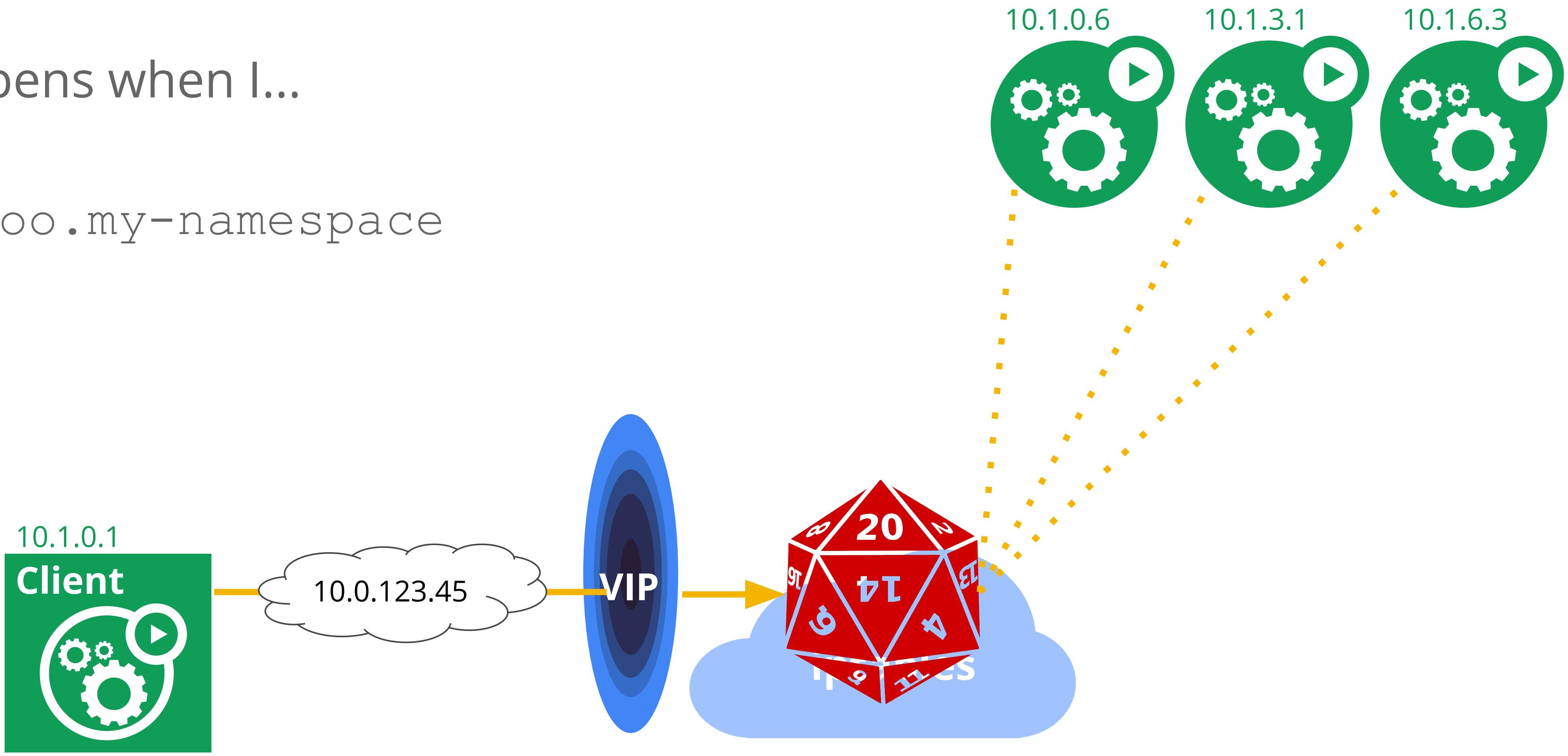
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

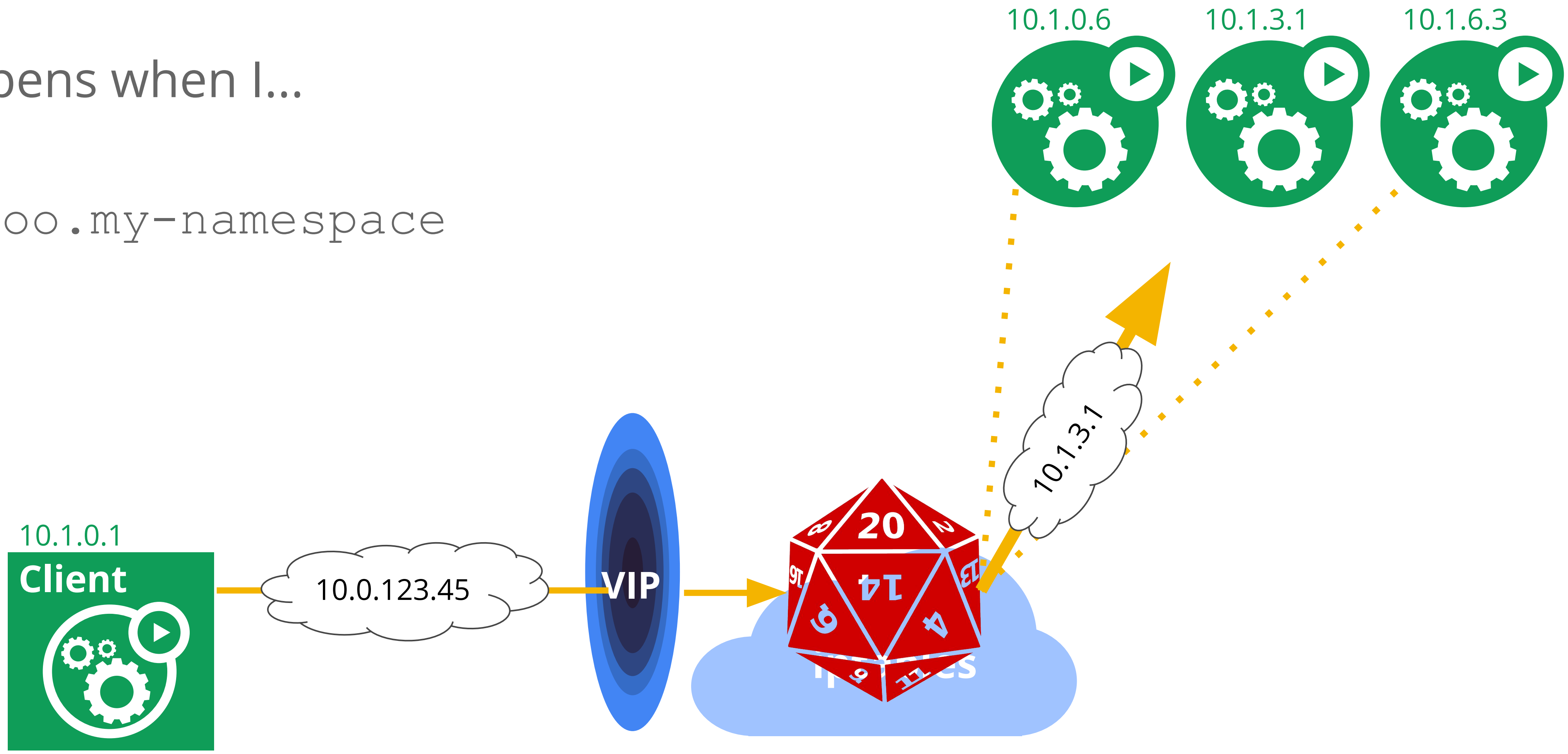
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

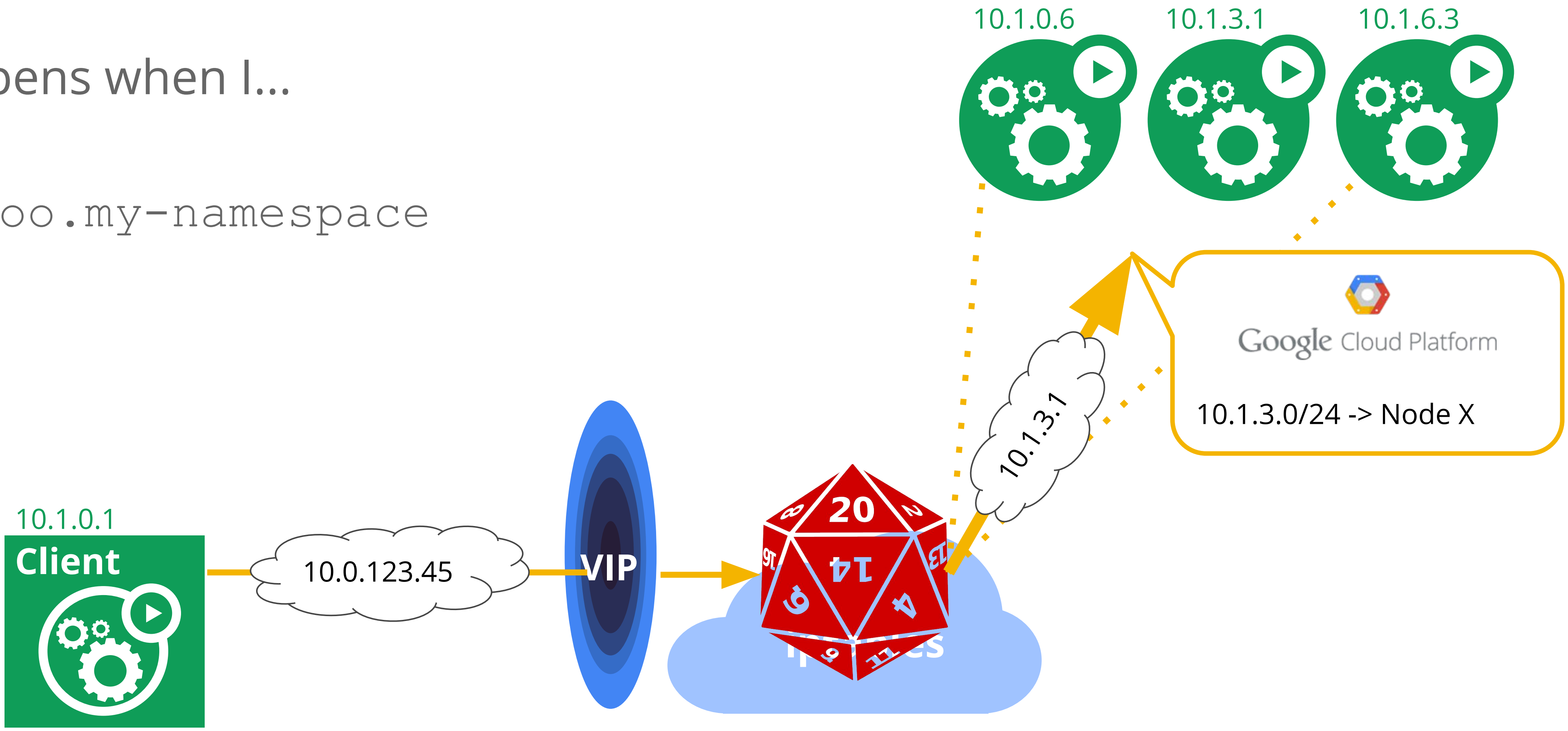
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I…

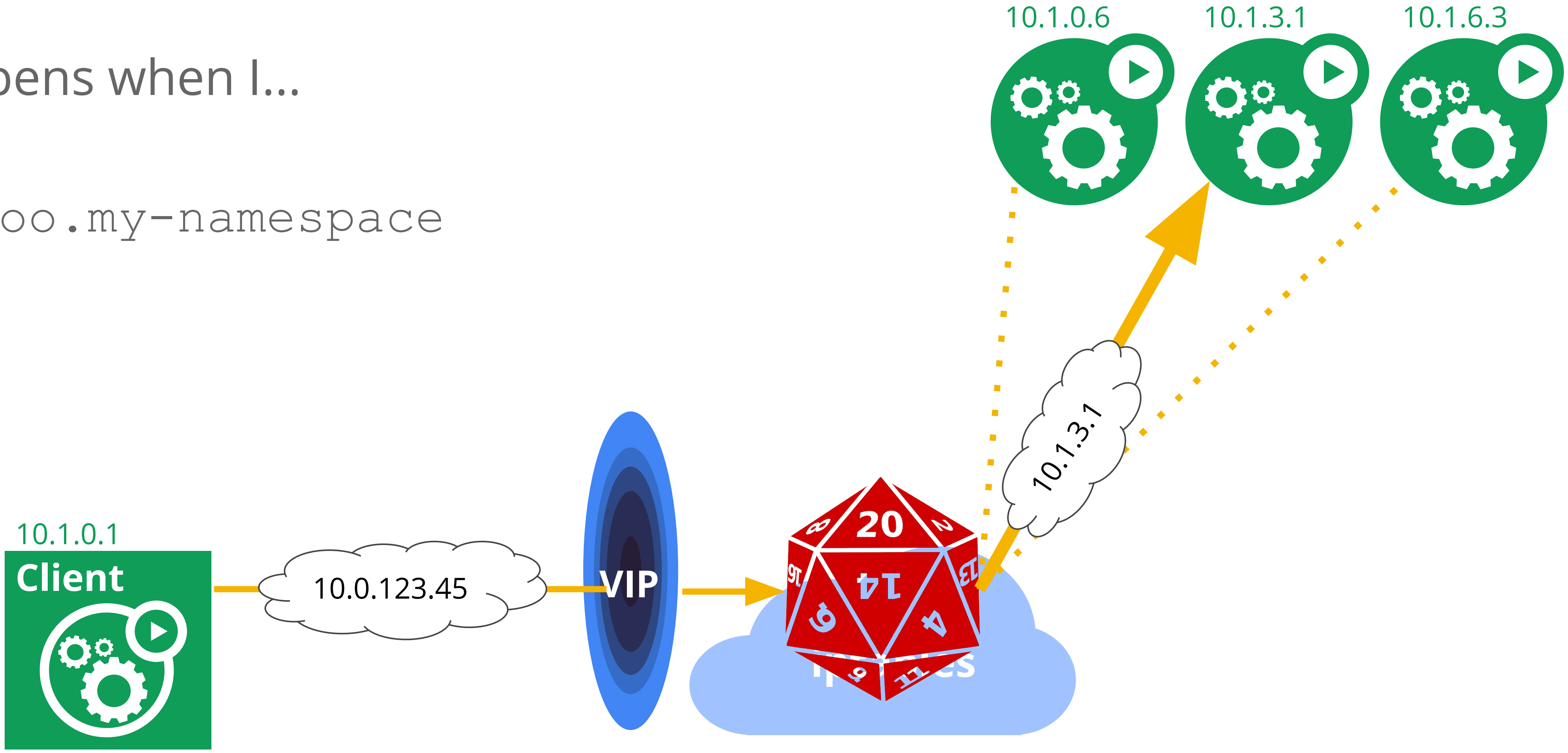
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

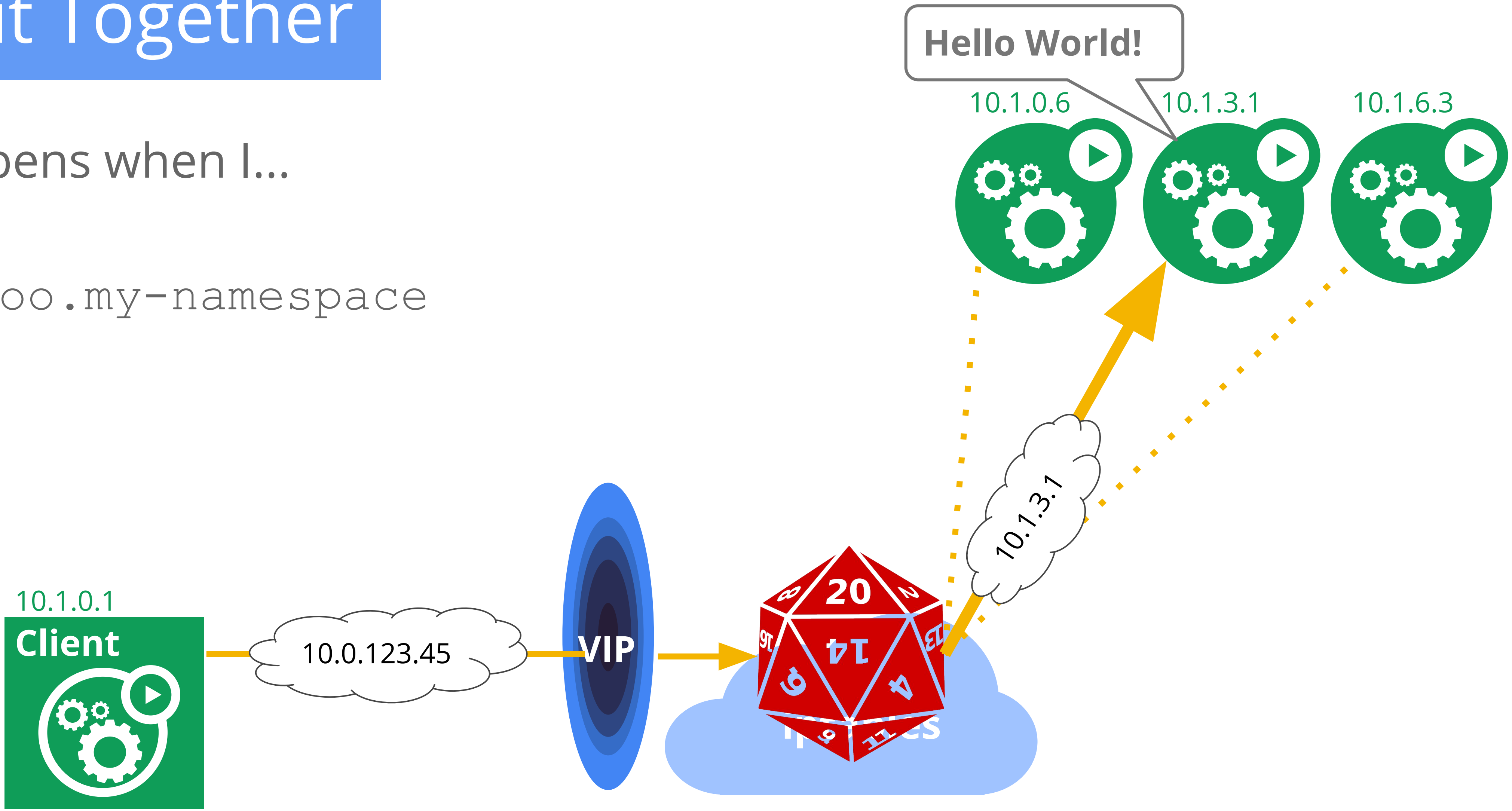
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

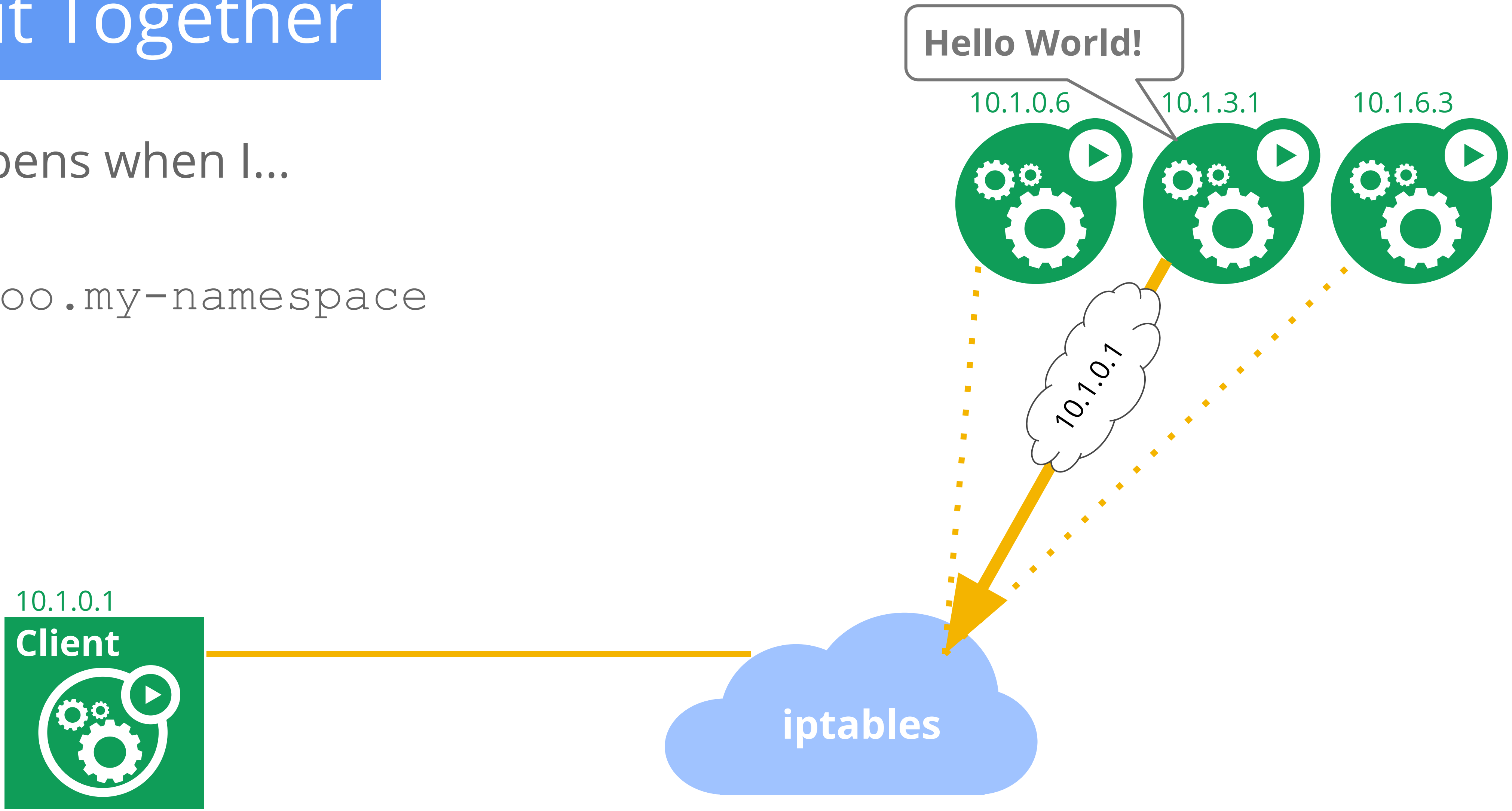
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I…

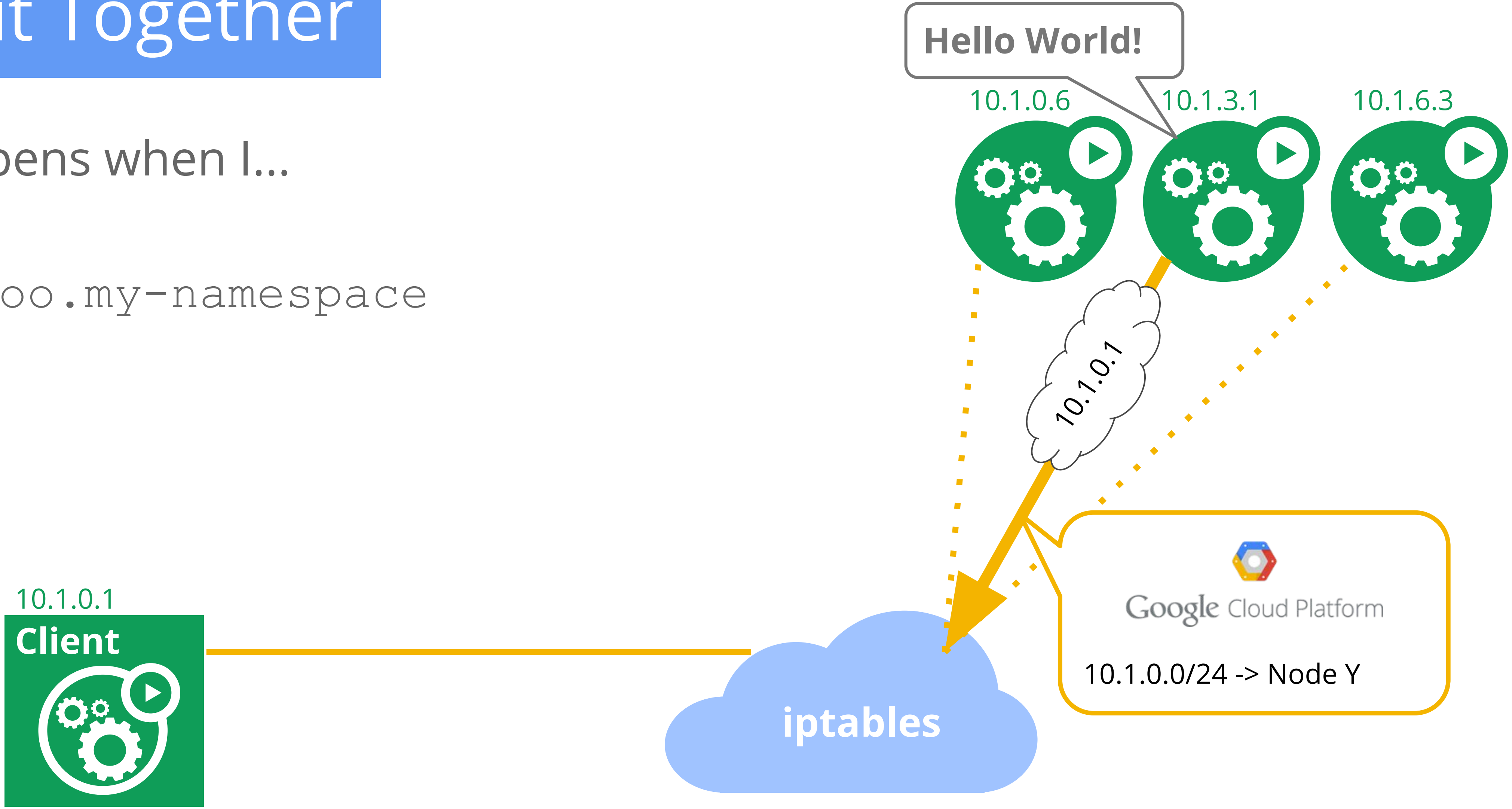
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

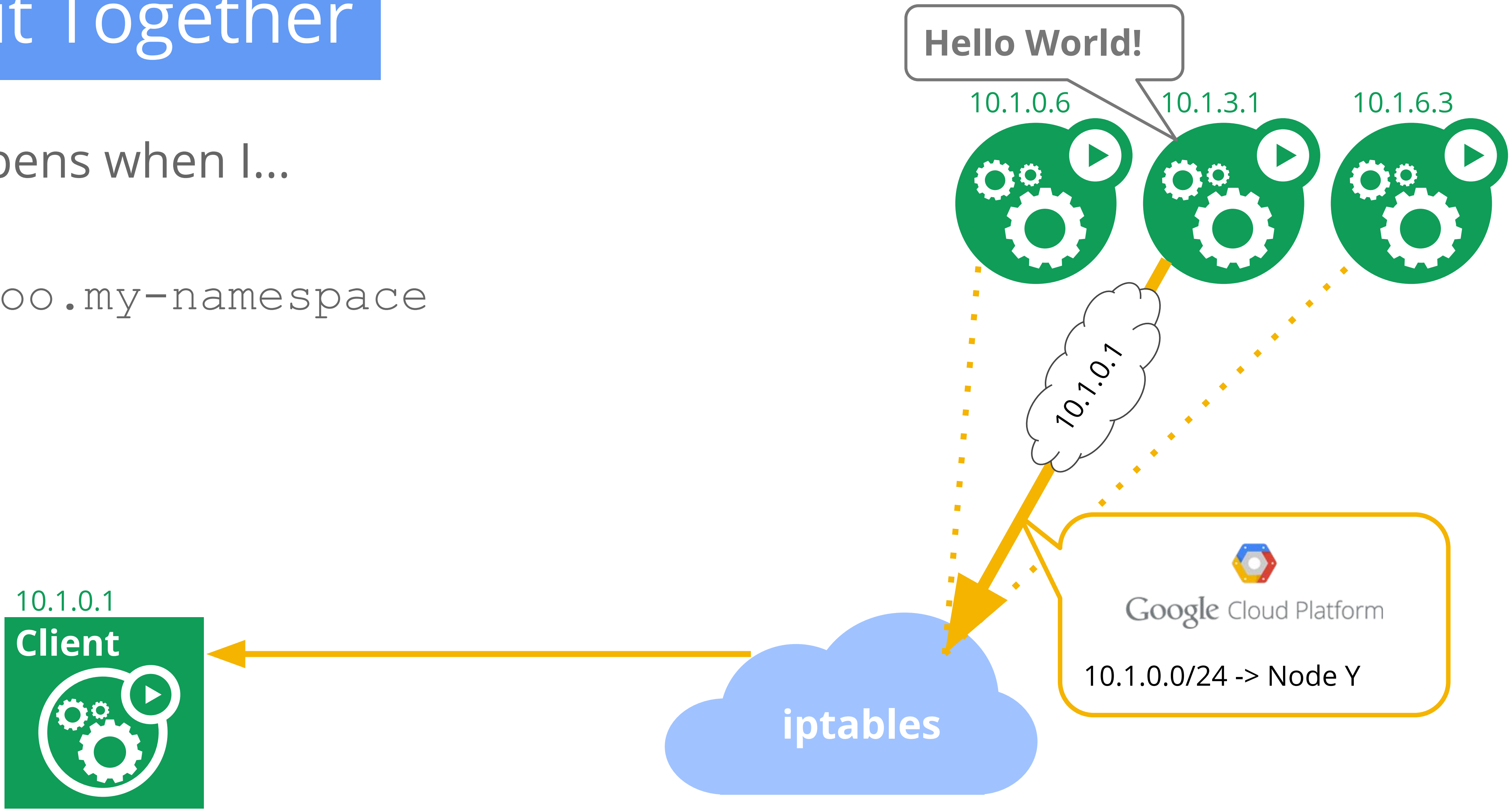
```
$ curl foo.my-namespace
```



Putting it Together

What happens when I...

```
$ curl foo.my-namespace
```

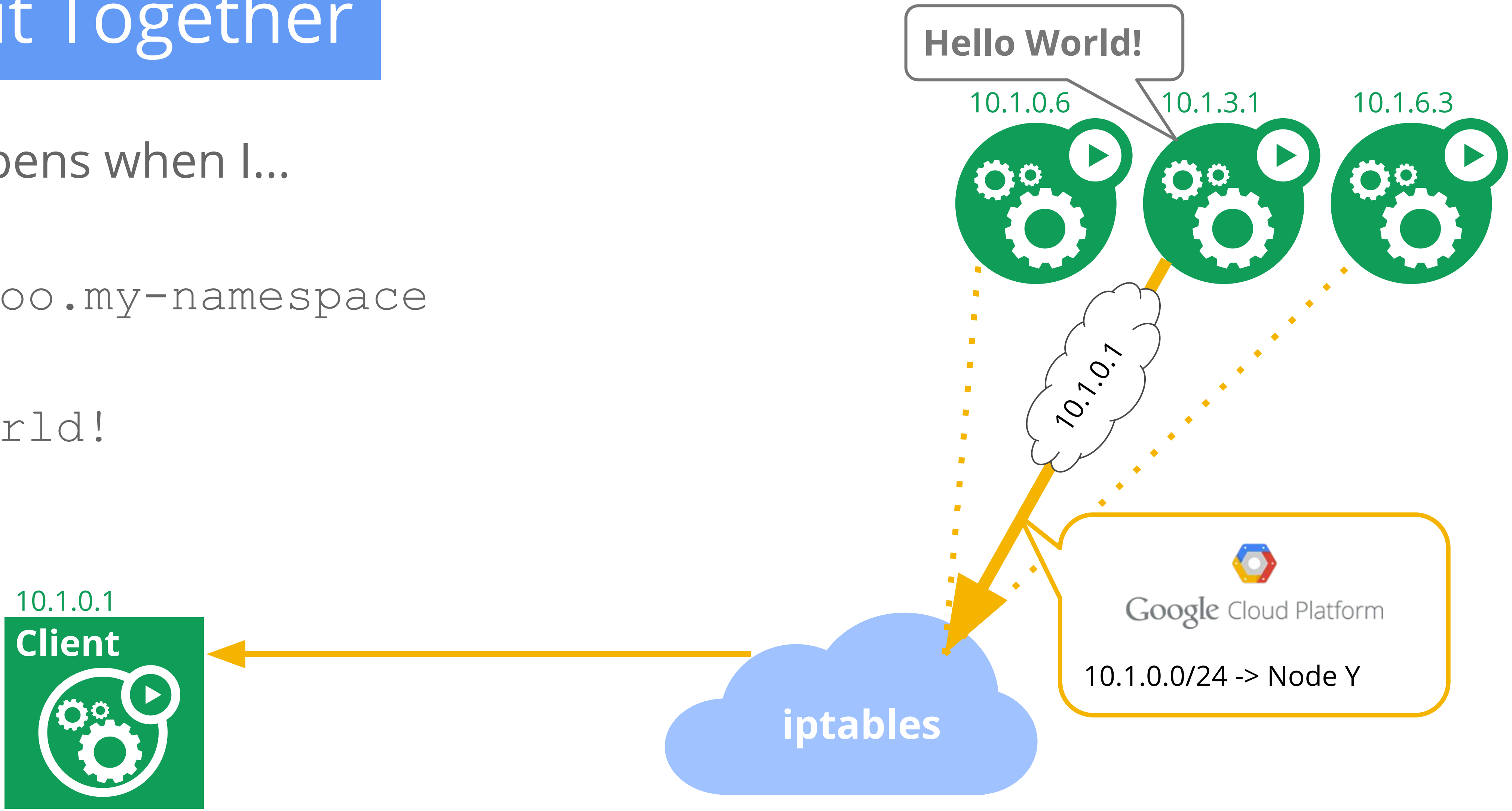


Putting it Together

What happens when I...

```
$ curl foo.my-namespace

Hello World!
```



What about external?

External Services

Services IPs are only available **inside** the cluster

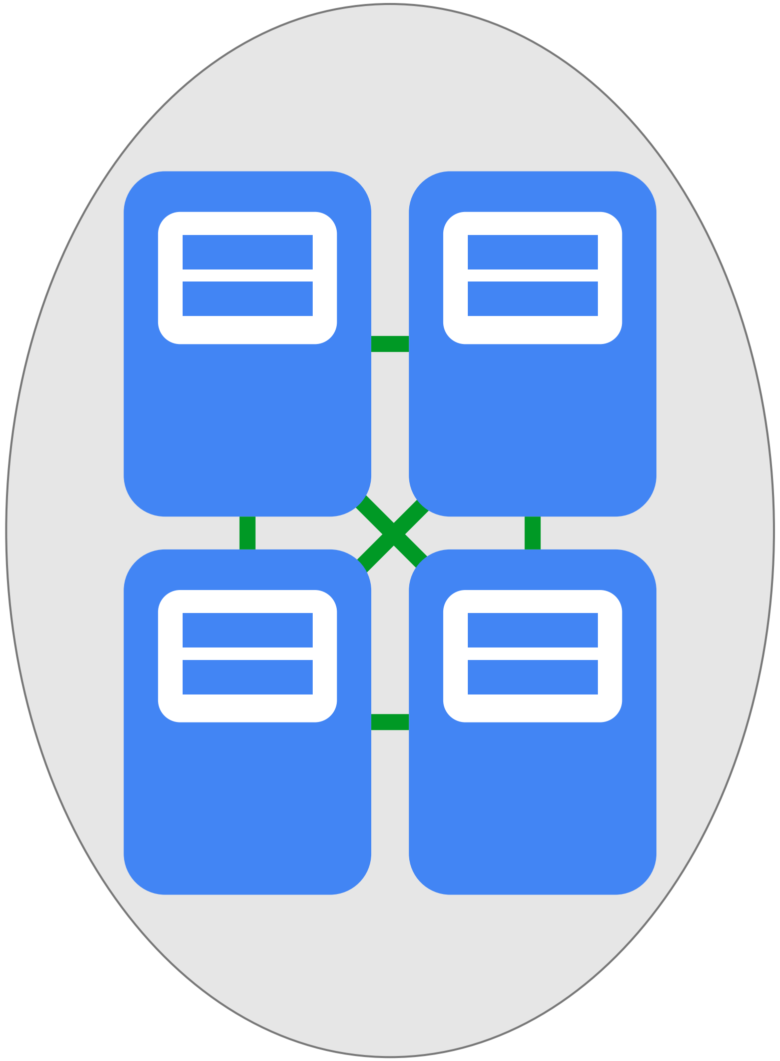
Need to receive traffic from “the outside world”

Builtin: Service “type”

- nodePort: expose on a port on every node
- loadBalancer: provision a cloud load-balancer

DiY load-balancer solutions

- socat (for nodePort remapping)
- haproxy
- nginx



The Bleeding Edge

Ingress (L7)

Services are assumed L3/L4

Lots of apps want HTTP/HTTPS

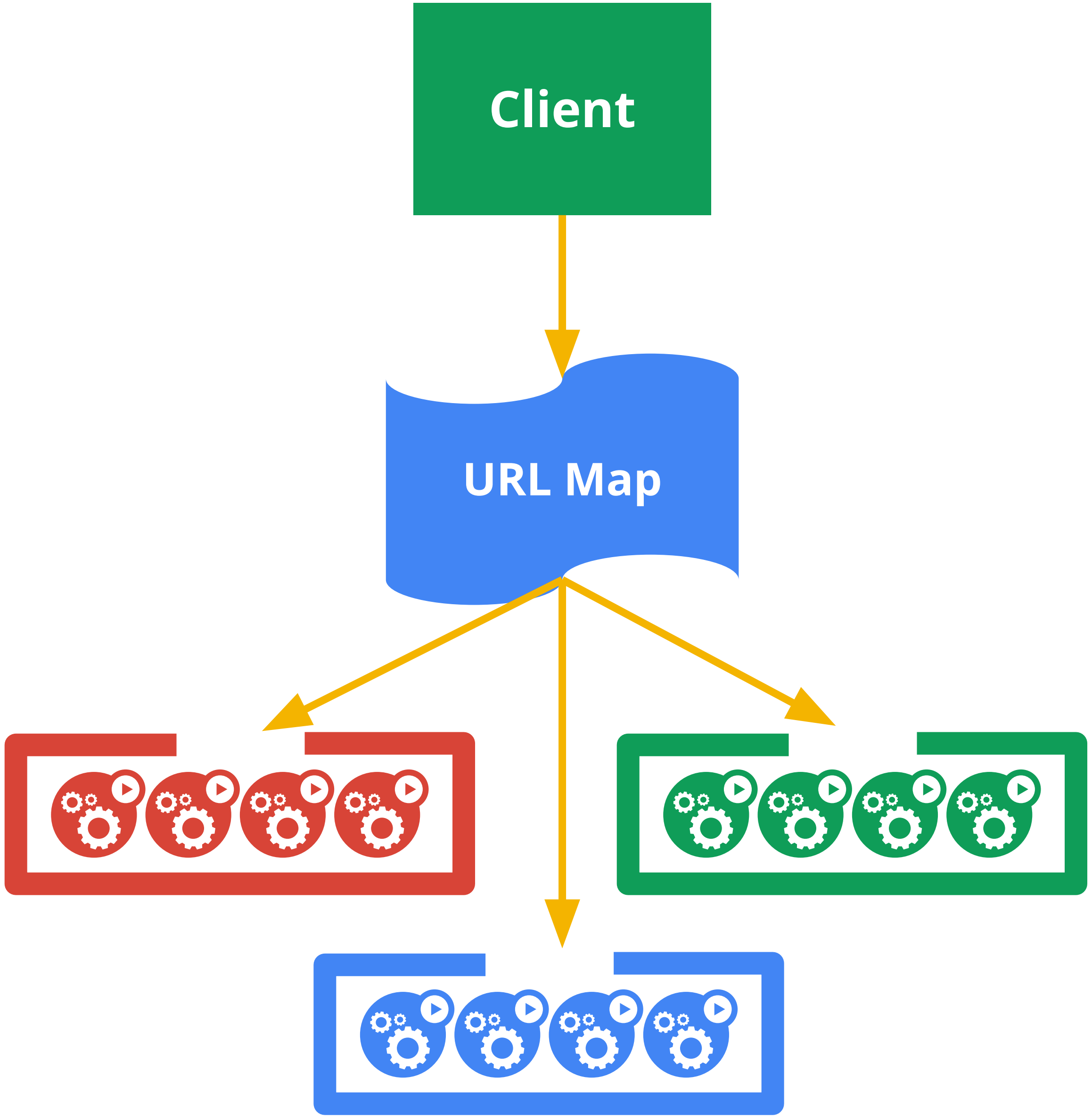
Ingress maps incoming traffic to backend services

- by HTTP host headers
- by HTTP URL paths

HAProxy and GCE implementations

No SSL yet

Status: **BETA** in Kubernetes v1.1



Ingress (L7)

Services are assumed L3/L4

Lots of apps want HTTP/HTTPS

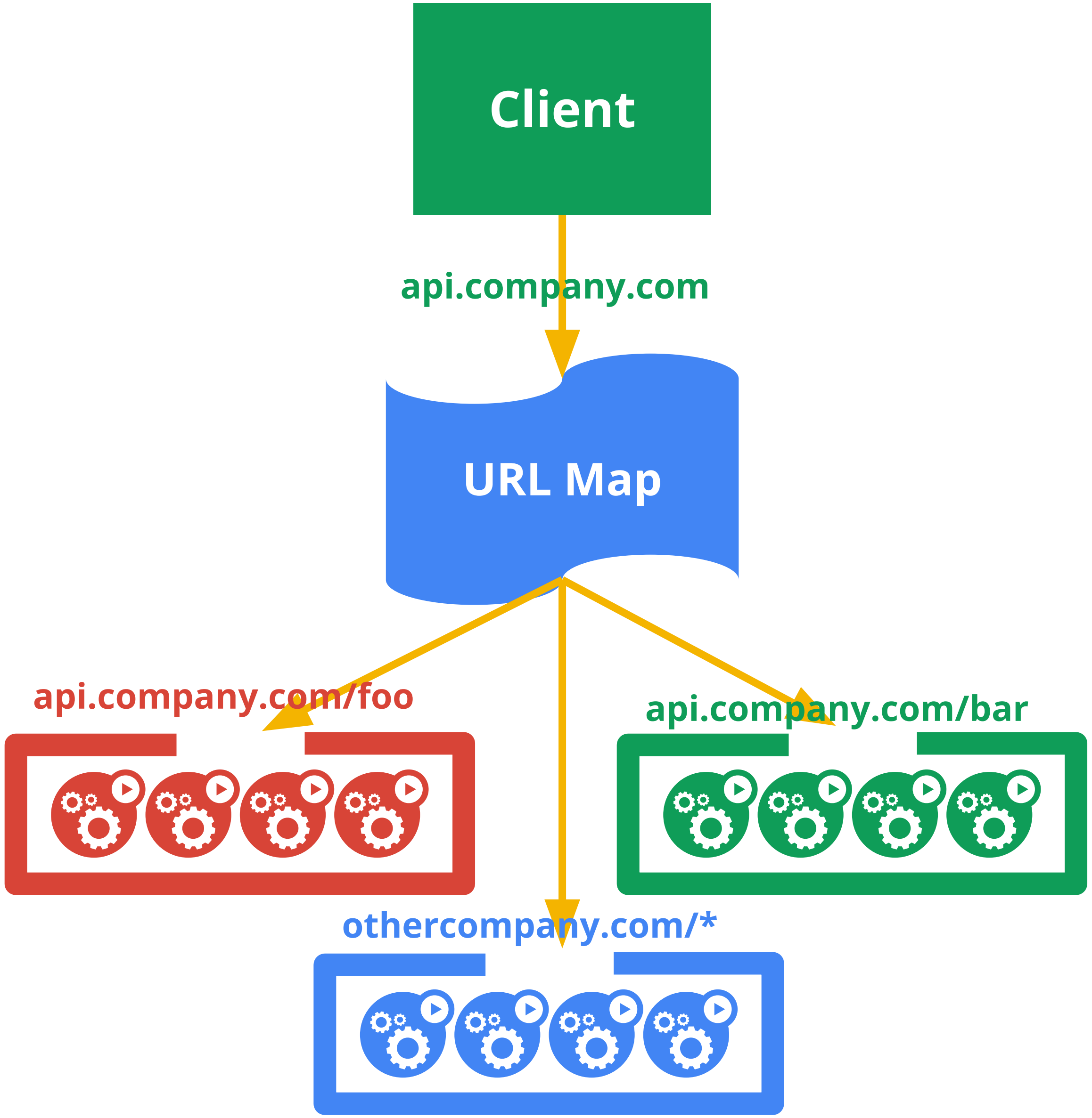
Ingress maps incoming traffic to backend services

- by HTTP host headers
- by HTTP URL paths

HAProxy and GCE implementations

No SSL yet

Status: **BETA** in Kubernetes v1.1



Network Plugins

Network Plugins

Introduced in Kubernetes v1.0

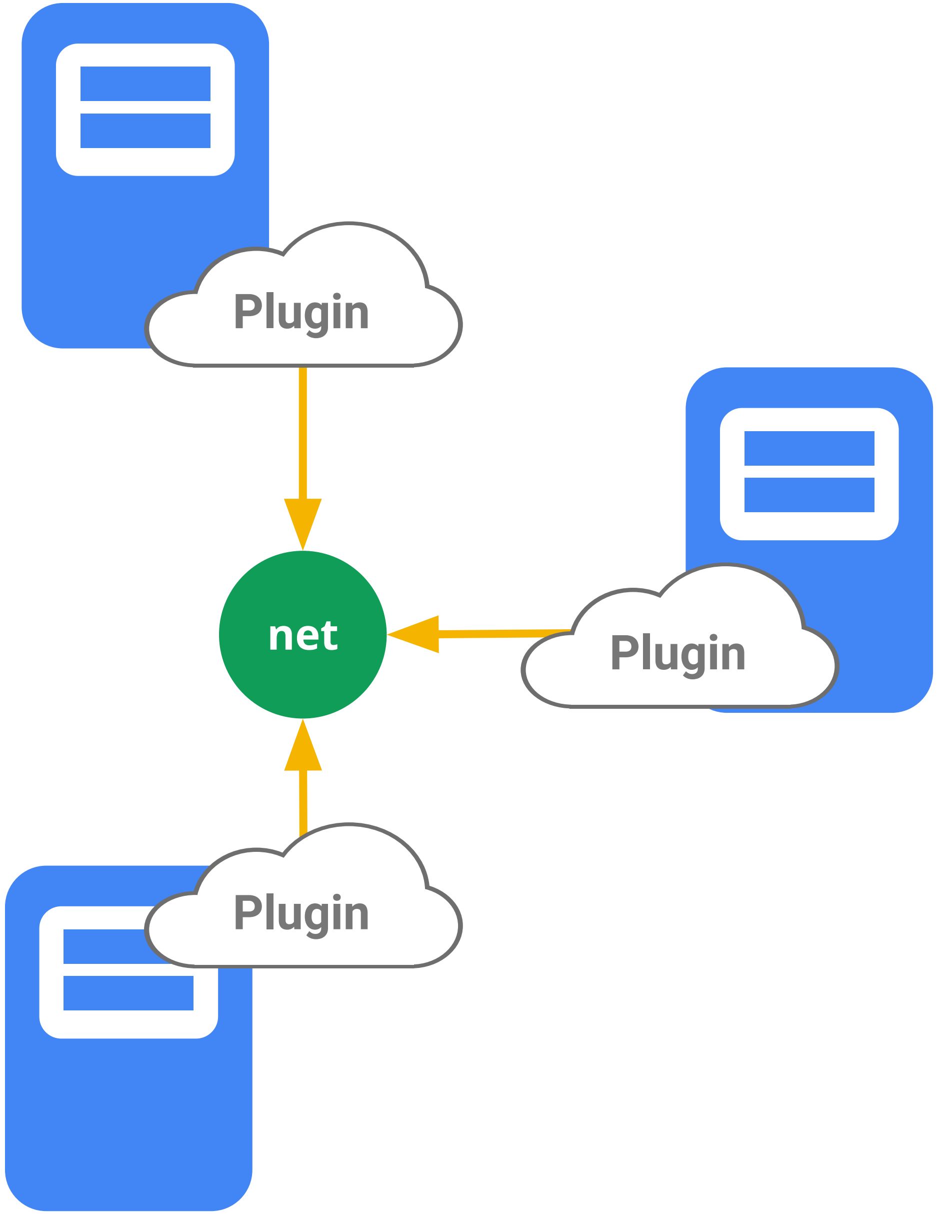
- VERY experimental

Uses **CNI** (CoreOS) in v1.1

- Simple exec interface
- Not using Docker libnetwork
  - but can defer to Docker for networking

Cluster admins can customize their installs

- DHCP, MACVLAN, Flannel, custom



Kubernetes is **Open**

- open community
- open design
- open source
- open to ideas

Networking is **Hard**

- help guide us!

http://kubernetes.io

https://github.com/kubernetes/kubernetes

slack: kubernetes twitter: @kubernetesio