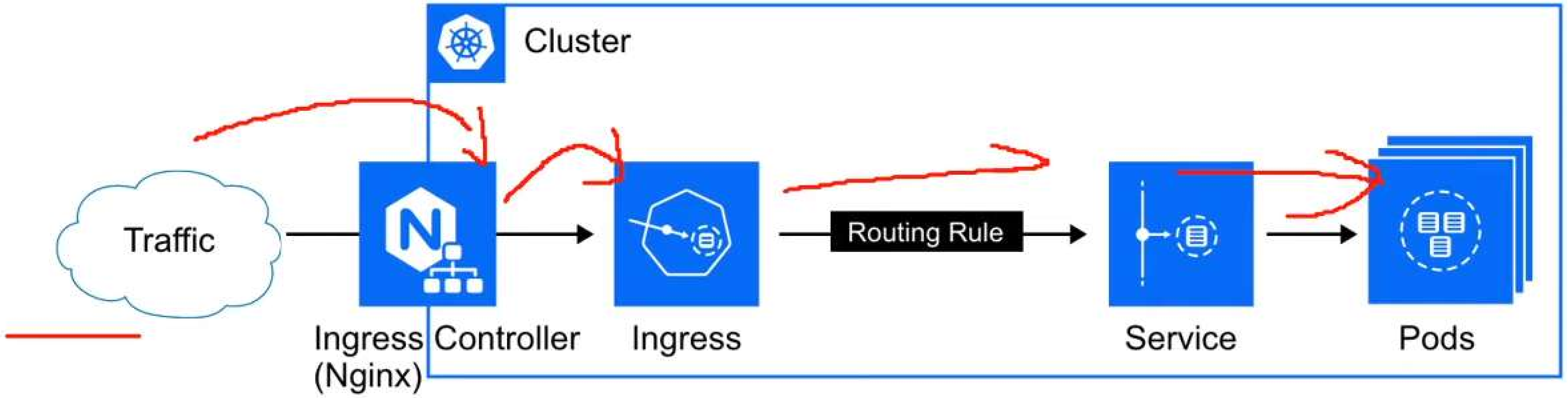
Kubernetes Ingress

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

The reason we use a K8s Ingress is so we can translate a custom domain on SSL to a service running within our K8s cluster.

In order for Ingress to work you need to use an Ingress Controller eg. AWS, GCE, Nginx



Ingress enables you to consolidate the traffic-routing rules into a single resource and runs as part of a Kubernetes cluster.

DNS

What is a Domain Name System (DNS)?

It is a service that is responsible for **translating (or resolving) a service name to its IP address**.

Service Registry	
frontend	10.90.142.14
backend	10.90.52.46
admin	10.90.41.101
...	...

CoreDNS is the default DNS server for Kuberentes and **ensures pods and services** haves Fully Qualified Domain Name (FQDN). Without CoreDNS the cluster communication would cease to work.

What is a FQDN?

a domain name that specifies its exact location in the tree hierarchy, also known as an absolute domain.

DNS

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

Lets take a look at the functionality provided via CoreDNS plugins

In-Tree Plugins (Internal Plugins)

- acl - enforces access control policies on source ip and prevents unauthorized access to DNS servers
- any - any gives a minimal response to ANY
- azure - enables serving zone data from Microsoft Azure DNS service
- cache - enables a frontend cache
- health - enables a health check endpoint
- log - enables query logging to standard output
- and many more…

Out-of-Tree Plugins (External Plugins)

- git – pull git repositories
- alias – replaces zones apex CNAMEs
- redisc – enables a networked cache using Redis
- Kubernetai – serve multiple Kubernetes within a Server
- and many more …..

DNS

CoreDNS pods are abstracted by a service object called kube-dns.

```
kubectl get service kube-dns -n kube-system
```

Each pod (any pod, not just CoreDNS pods) has a **resolv.conf** file to help with DNS resolving.

```
kubectl exec -it my-pod -- sh
cat /etc/resolv.conf
# nameserver 10.100.0.10
# search default.svc.cluster.local svc.cluster.local cluster.local ec2.internal
# options ndots:5
```

nslookup can be used to see what resolves Stored with CoreDNS

```
nslookup my-service-name
# Server:     10.100.0.10
# Address:    10.100.0.10#53
# Name:       kubernetes.default.svc.cluster.local
# Address:    10.100.0.1
```

Load Balancing

What is Load Balancing?
Load Balancing is a networking component, where traffic flows through the load balancer, and the load balancer decides how to distribute the traffic to multiple targets eg. (compute nodes) based on a set of rules.

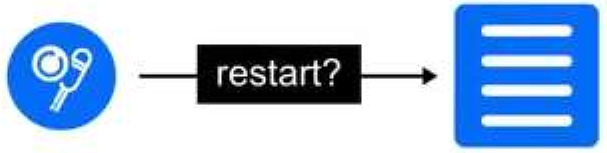
Ingress and **Service** K8s both have load balancing.

External Load Balancer	Ingress	Service	Internal Load Balancing (iptables / ipvs)
Load balancing control By third-party service	load balancing algorithm backend weight scheme	persistent sessions dynamic weights	Load balancing to containers within pods. Randomly distributed

Probes

Probes are used to detect the state of a container

Liveness Probe

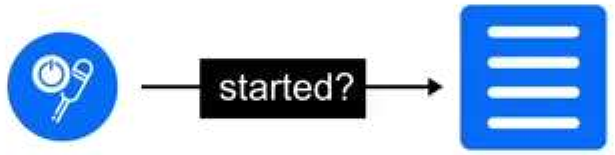


The kubelet uses liveness probes to know **when to restart a container.**
For example, liveness probes could catch a deadlock, where an application is running, but unable to make progress. Restarting a container in such a state can help to make the application more available despite bugs.

Readiness Probe

The kubelet uses readiness probes to know **when a container is ready to start accepting traffic.**
A Pod is considered ready when all of its containers are ready.
One use of this signal is to control which Pods are used as backends for Services.
When a Pod is not ready, it is removed from Service load balancers.

Startup Probe



The kubelet uses startup probes to know **when a container application has started.**
If such a probe is configured, it disables liveness and readiness checks until it succeeds, making sure those probes don't interfere with the application startup.
This can be used to adopt liveness checks on slow starting containers, avoiding them getting killed by the kubelet before they are up and running.

Netfilter

netfilter
firewalling, NAT, and packet mangling for linux

The netfilter project enables:

- packet filtering
- network address and port translation (NAPT)
- Translation packet logging
- Userspace packet queueing
- other packet mangling

The netfilter hooks are a framework inside the Linux kernel that allows kernel modules to register callback functions at different locations of the Linux network stack.

The registered callback function is then called back for every packet that traverses the respective hook within the Linux network stack.

Projects build ontop of Netfilter:

- Iptables — generic firewalling software that allows you to define rulesets
- Nftables – successor to iptables, more flexible, scalable and performance packet classification
- IPVS — specifically designed for load balancing, uses hash mapping

IP Tables

What is a Userspace?

Modern computer operating system segregates virtual memory into kernel space and user space
Kernel space is reserved for running a privileged operating system kernel, kernel extensions, and device drivers.
User space is the memory area where application software and some drivers execute.

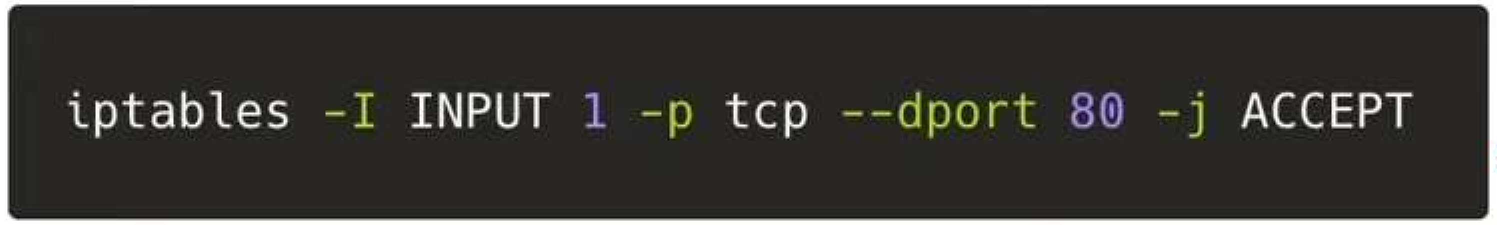
What is IP Tables?

iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall

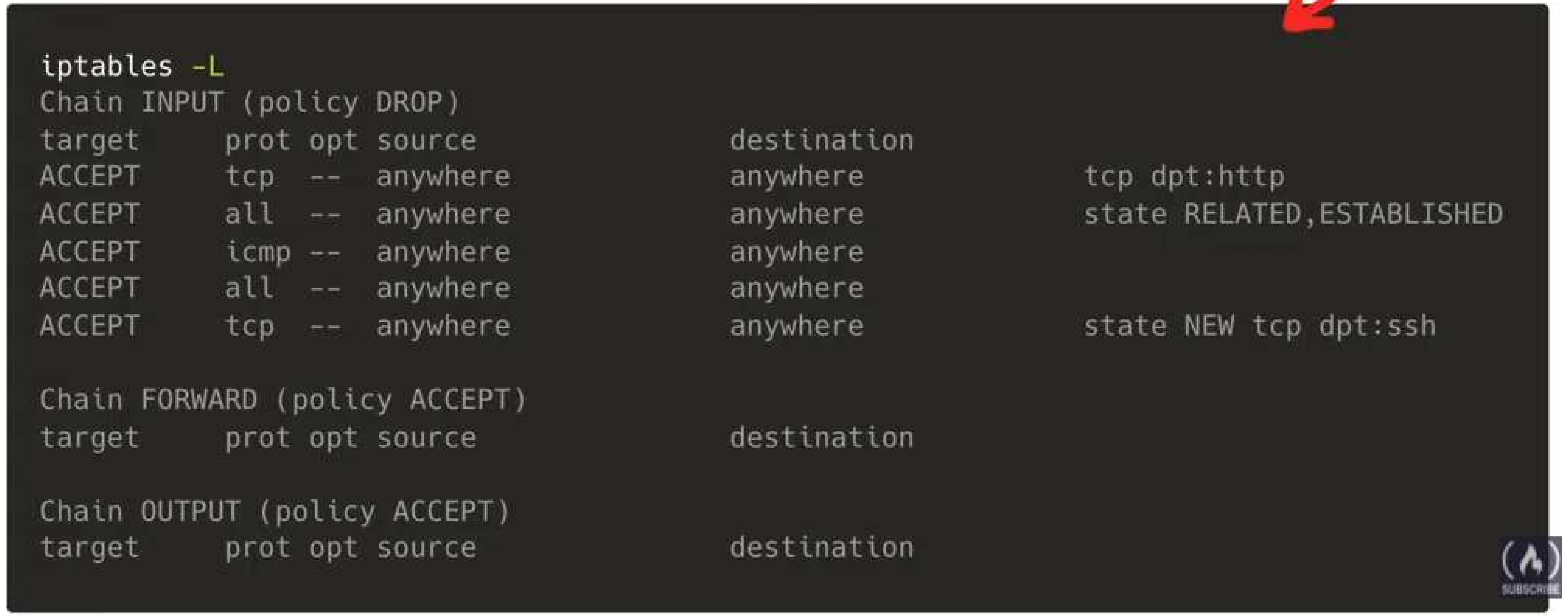
- *iptables* applies to IPv4
- ip6tables to IPv6

Iptables are simply virtual firewalls on Linux

It is common to and modify iptables to restrict access based on ports and protocols



```
iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
```



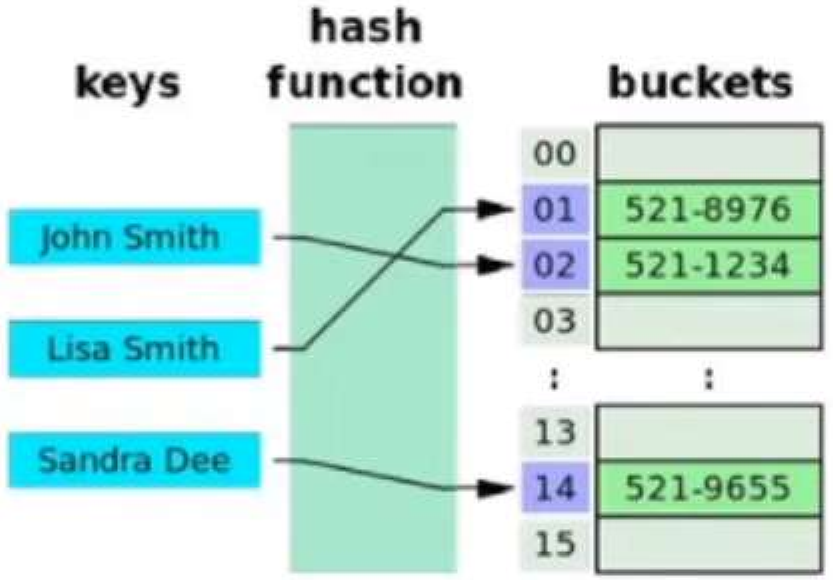
```
iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

IPVS

IP Virtual Server (IPVS) uses the NetFilter framework.
IPVS also incorporates LVS (Virtual Linux Server)

Iptables struggles to scale to tens of thousands of services as Iptables is bottlenecked at 5000 nodes per cluster

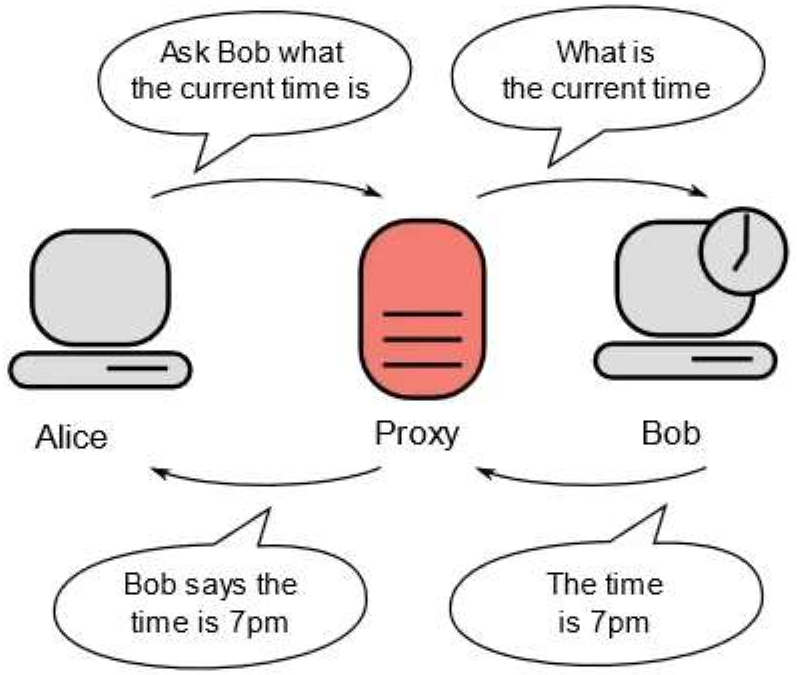
IPVS is specifically designed for load balancing and uses more efficient data structures (hash tables) allowing for almost unlimited scale under the hood



In the future the Kube Proxy will default to using IPVS.

Various Proxies

What is a proxy?
a server application that acts as an intermediary between a client requesting a resource and the server providing that resource



There are **many kinds of proxies you will encounter** in Kubernetes:

- **Kubectl proxy** — proxies from a localhost address to the Kubernetes apiserver
- **Apiserver proxy** — a bastion built into the apiserver, connects a user outside of the cluster to cluster IPs which otherwise might not be reachable
- **Kube proxy** — runs on each node and used to reach services
- **Proxy/Load balancer in front of API servers** — acts as load balancer if there are several apiserver
- **Cloud Load Balancers** — for external cluster traffic to reach pods

- **Forward Proxy:** A bunch of servers egressing traffic have to pass through the proxy first
- **Reverse Proxy:** Ingress traffic trying to reach a collection of servers

Kubernetes Proxy

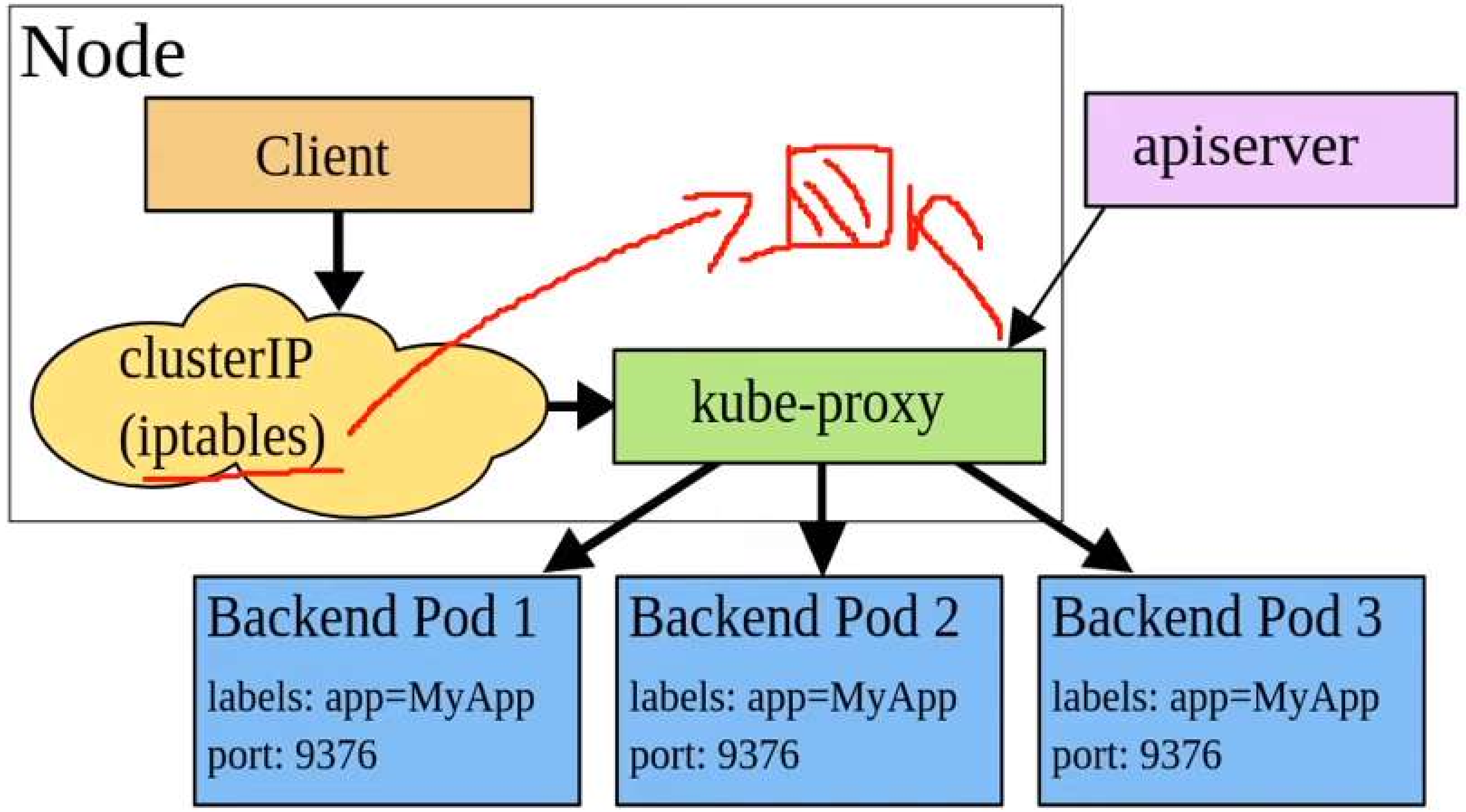
kube-proxy is a network proxy that **runs on each node** in your cluster. It is designed to load **balance traffic to pods**.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer (if there is one and it's available). Otherwise, kube-proxy forwards the traffic itself.

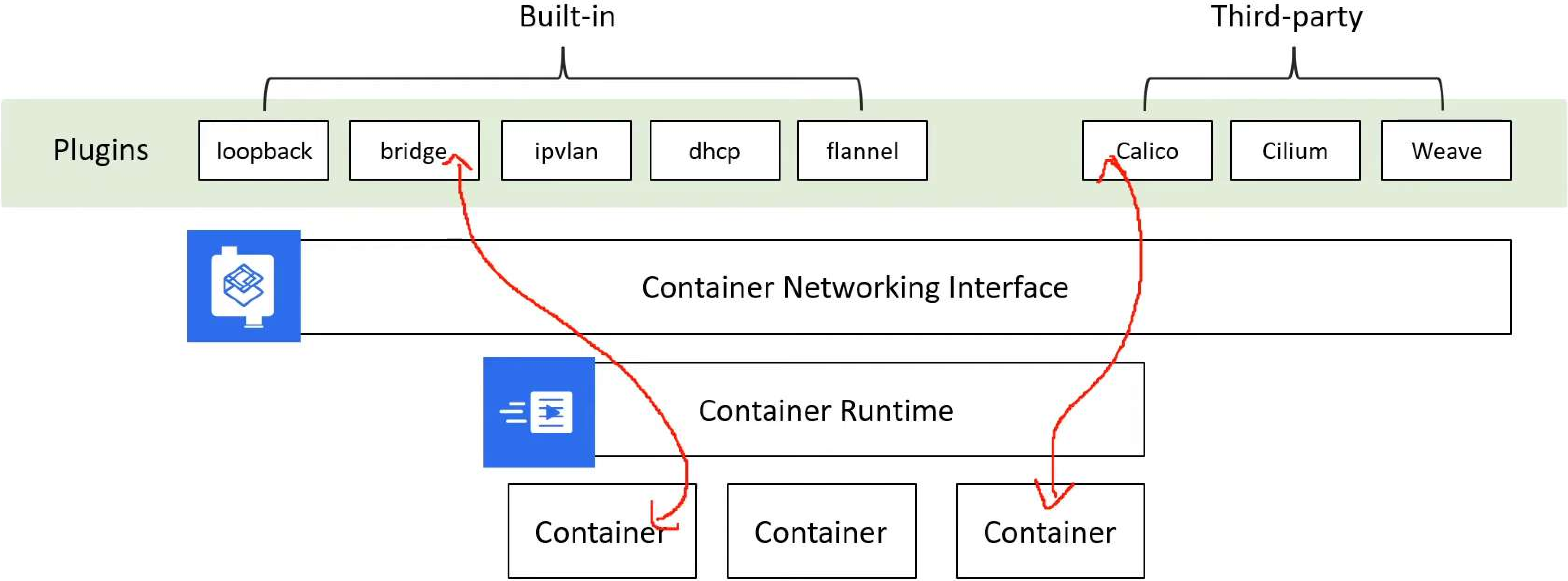
Kube-proxy can run in three modes:

1. **iptables** (default). — Suited for simple use cases
2. **Ipvs** — Suites for 1000+ services.
3. **Userspace** (legacy) — Not recommended for use



Container Networking Interface (CNI)

The **Container Networking Interface (CNI) is** a specification (open standard) for writing plugins to configure networking interfaces for linux containers



Service Mesh

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

You don’t have to run a service mesh, but in most production use-cases it is recommended to do so.

Available Service Meshes for Kubernetes

Istio is the currently most popular service mesh for Kubernetes due to its highly configurable nature.
Istio uses Envoy as its proxy. Istio is a not a CNCF project.

Envoy is an open-source edge and service proxy. Multiple Service Meshes uses Envoy as its proxy.
Envoy is a graduated CNCF project.

Kuma is a CNCF sandbox project that uses Envoy as its proxy.

Linkerd is a CNCF graduated project is known for having strong security and ”it just works”.
Linkerd does not use Envoy, instead it uses a simple and ultralight “micro-proxy” called Linkerd2-proxy

Consul is an open-source service mesh by Hashicorp. Its not a CNCF project.
Consul is offered by Hashicorp as managed cloud Service Mesh.

Envoy

Envoy is a self contained process that is designed to run alongside every application server.
Envoy can be installed on a Virtual Machine or as a Container.

Envoy supports a wide range of functionality

- L3/L4 filter architecture
- HTTP L7 filter architecture:
- First class HTTP/2 support
- HTTP/3 support
- HTTP L7 routing
- gRPC support
- Service discovery and dynamic configuration
- Health checking
- Advanced load balancing
- Front/edge proxy support
- Best in class observability

In practice you likely will not install and manually configure Envoy,
You would allow a Service Mesh control plane to install into your pods
A Service Mesh will may come with a UI or configuration files to configure your envoy