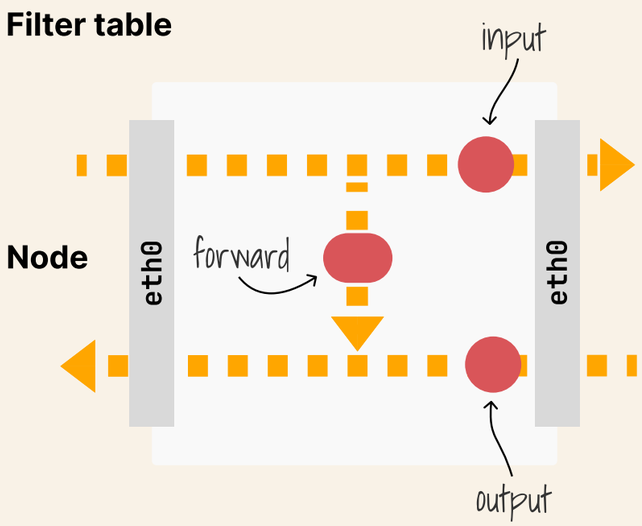
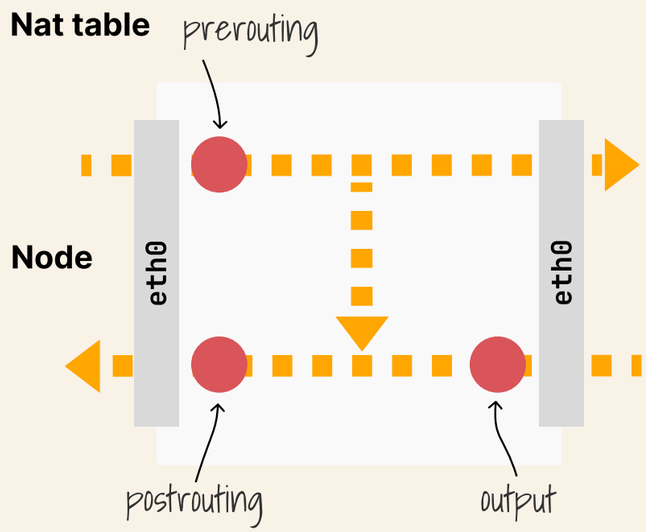
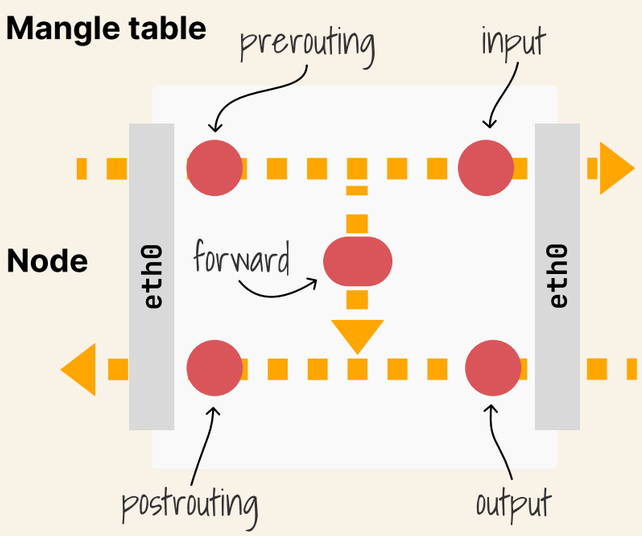
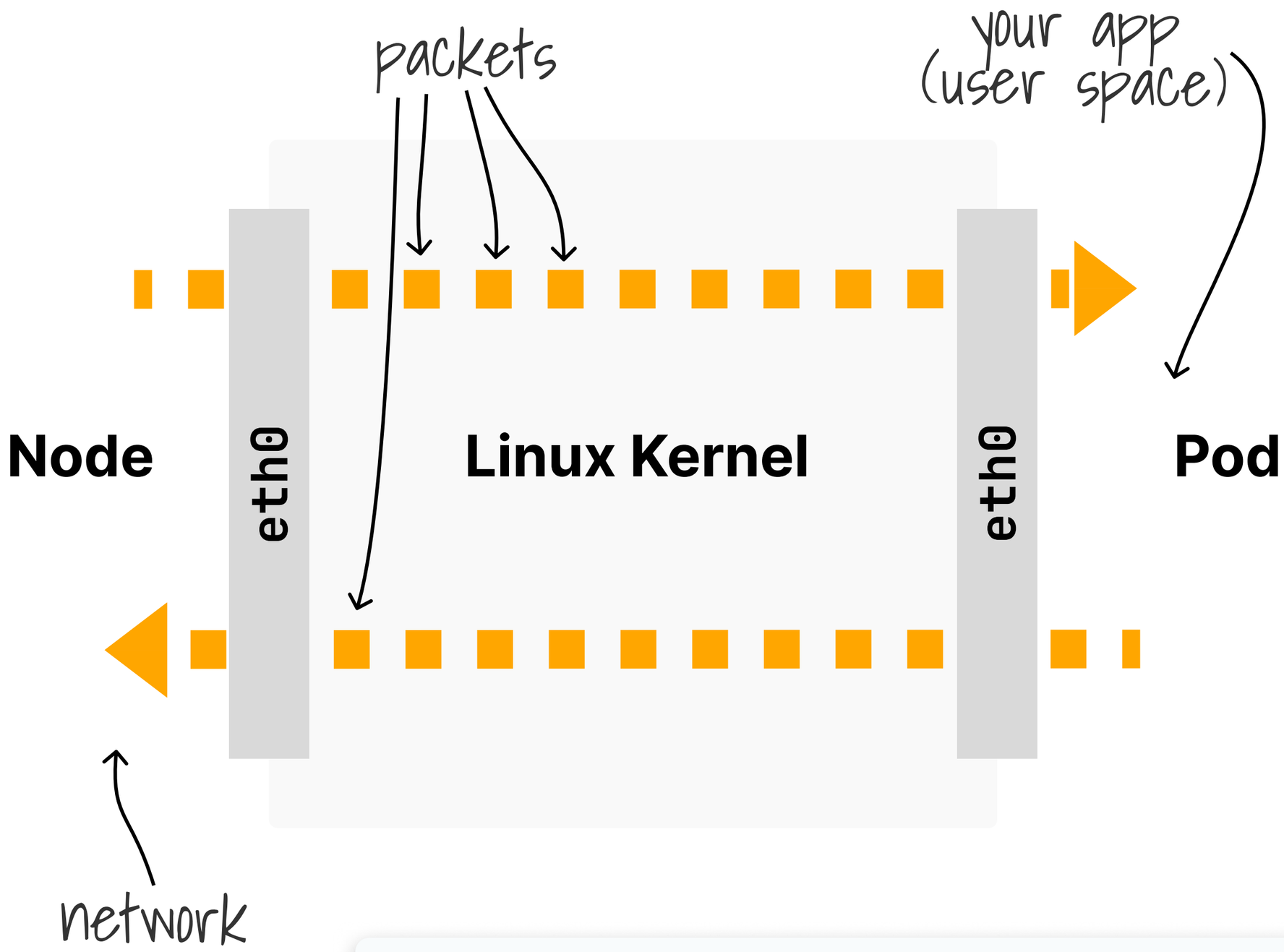
KUBE-PROXY AND IPTABLES EXPLAINED





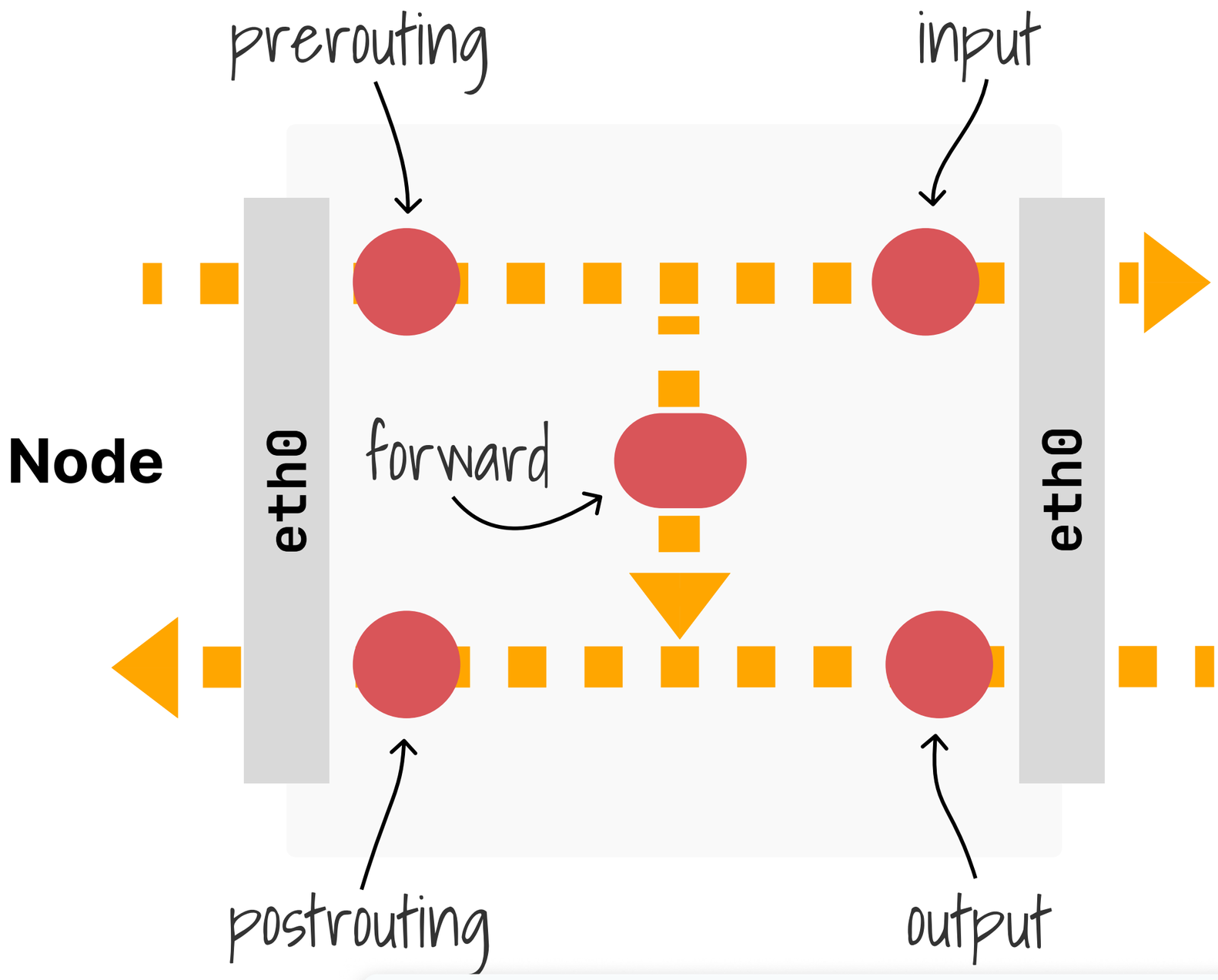


Daniele Polencic



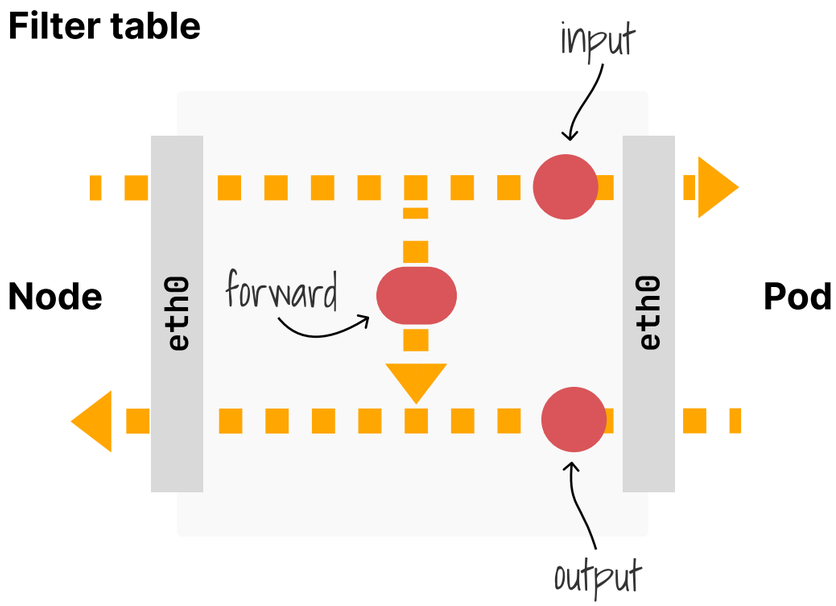
Kubernetes Services relies on the **Linux kernel's networking stack and the Netfilter framework to modify and redirect network traffic.**

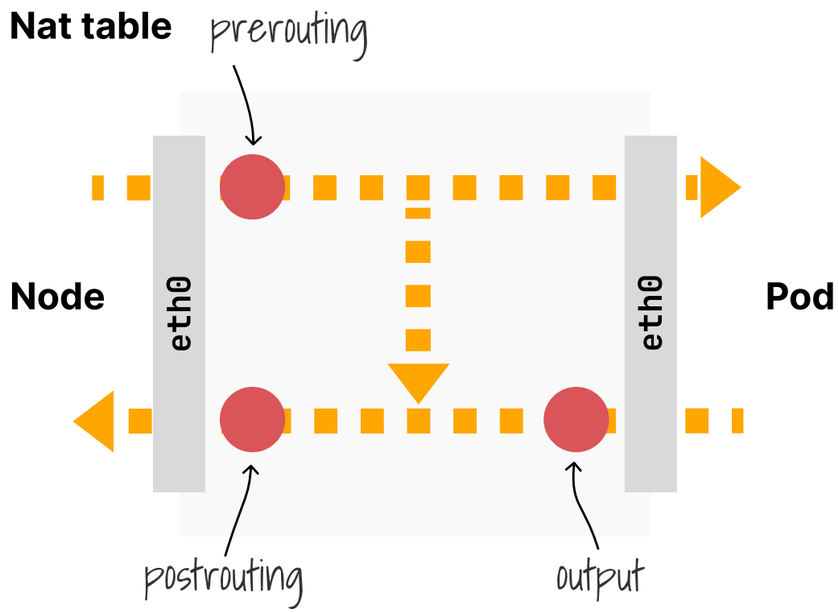
The Netfilter framework provides hooks at different stages of the networking stack where rules can be inserted to filter, change, or redirect packets.

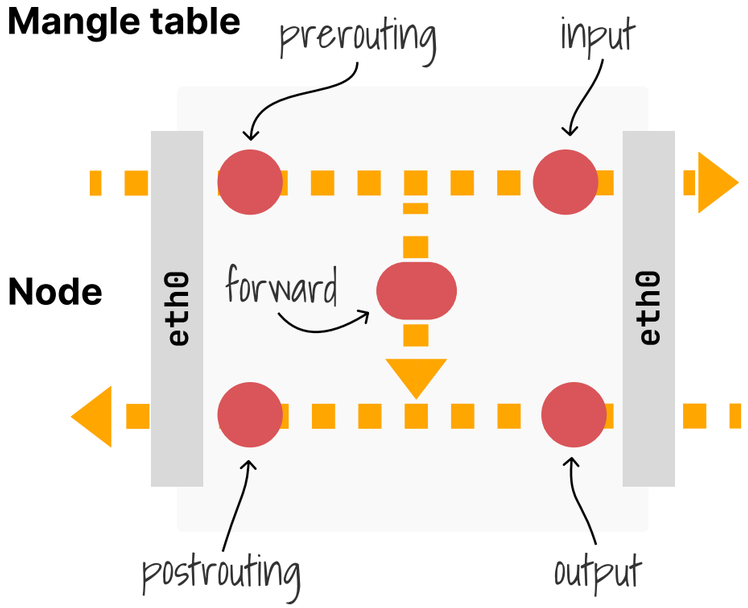


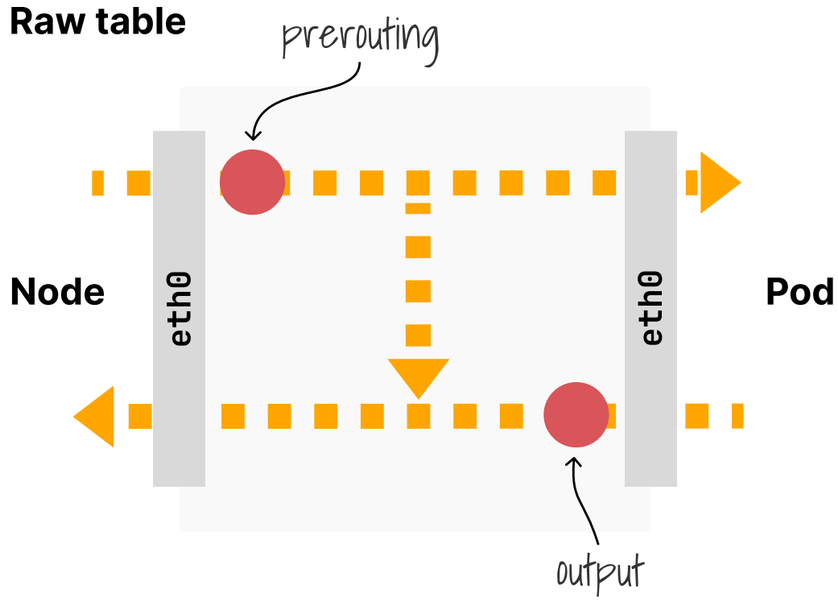
The Netfilter framework offers five hooks to modify network traffic: PRE_ROUTING, INPUT, FORWARD, OUTPUT, and POST_ROUTING.

These hooks represent different stages in the networking stack, allowing you to intercept and modify packets at various points in their journey.





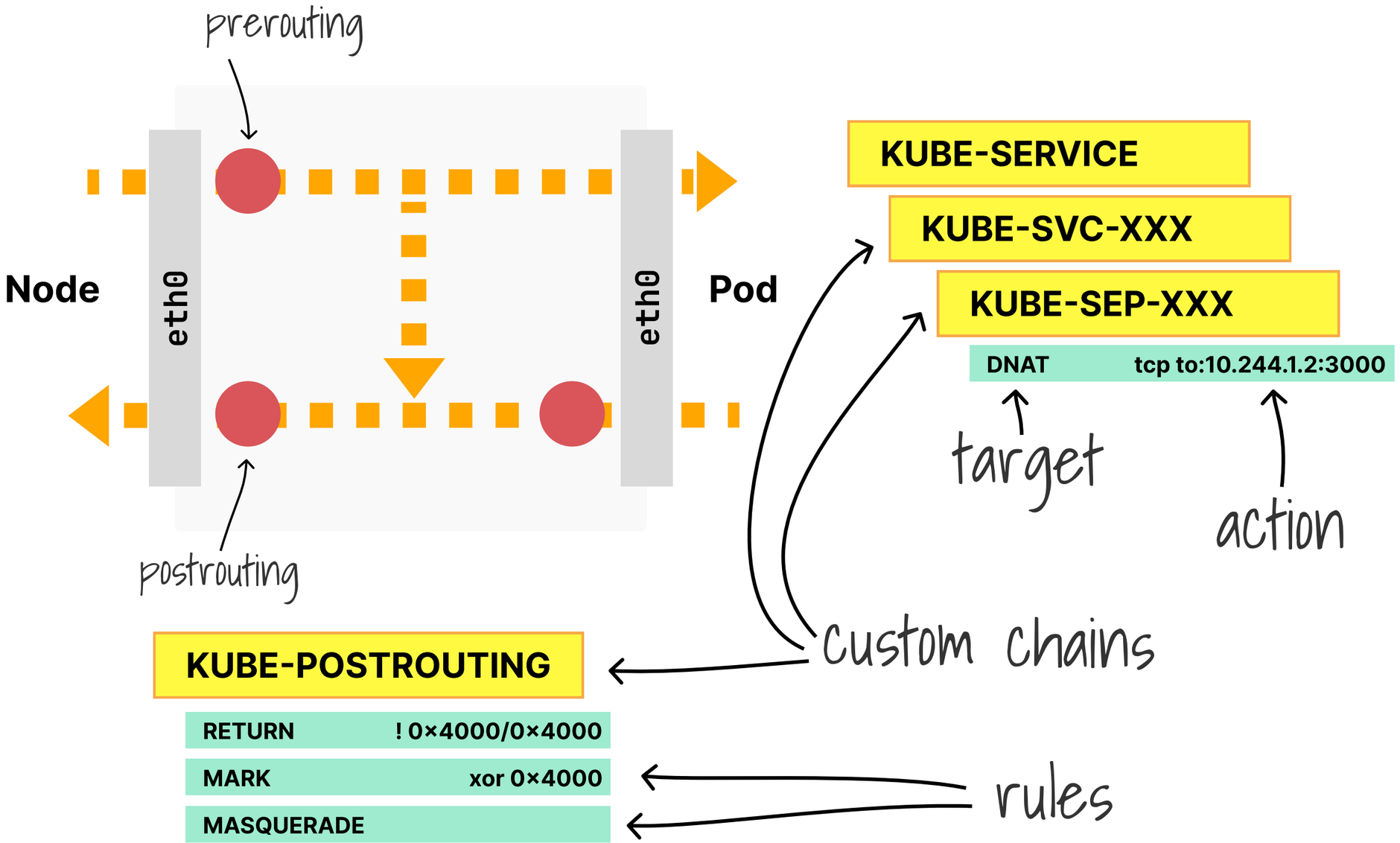




Iptables, a user-space binary, allows you to configure rules for these hooks.

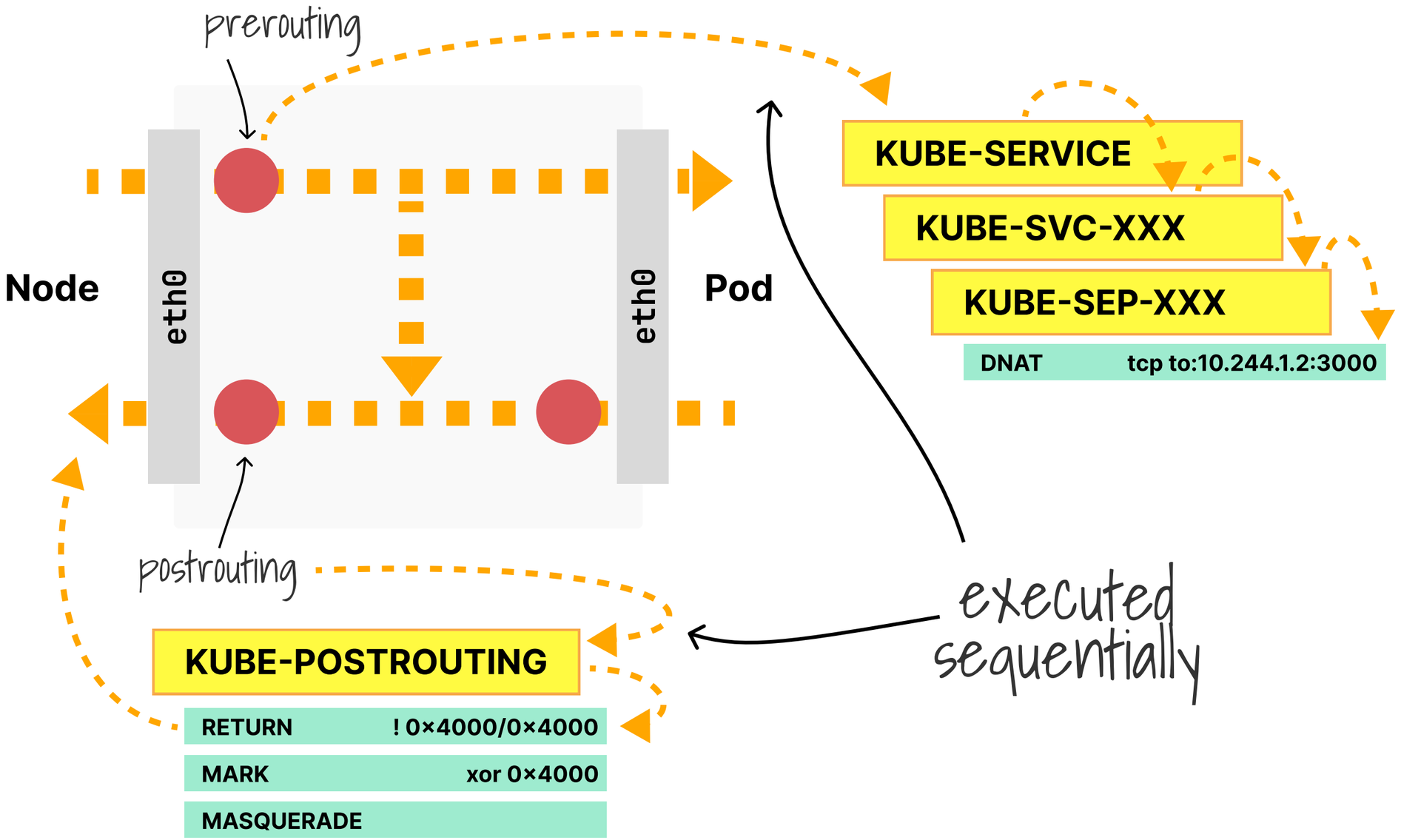
It has five tables: filter, nat, mangle, raw, and security.

Each table has different hooks available, and the rules within these tables can be used to filter, modify, or redirect network traffic.



Iptables modifies network traffic using chains, rules, targets, and actions.

Chains are sequences of rules that are executed sequentially. Rules define the conditions for matching packets, and targets specify the action to be taken when a rule matches.



Chains in iptables can be linked to "built-in" chains associated with the Netfilter hooks.

This allows you to create complex workflows by chaining multiple rules and linking them to different hooks in the networking stack.

In Kubernetes, iptables rules are invoked when you request a ClusterIP Service and the traffic reaches the node.

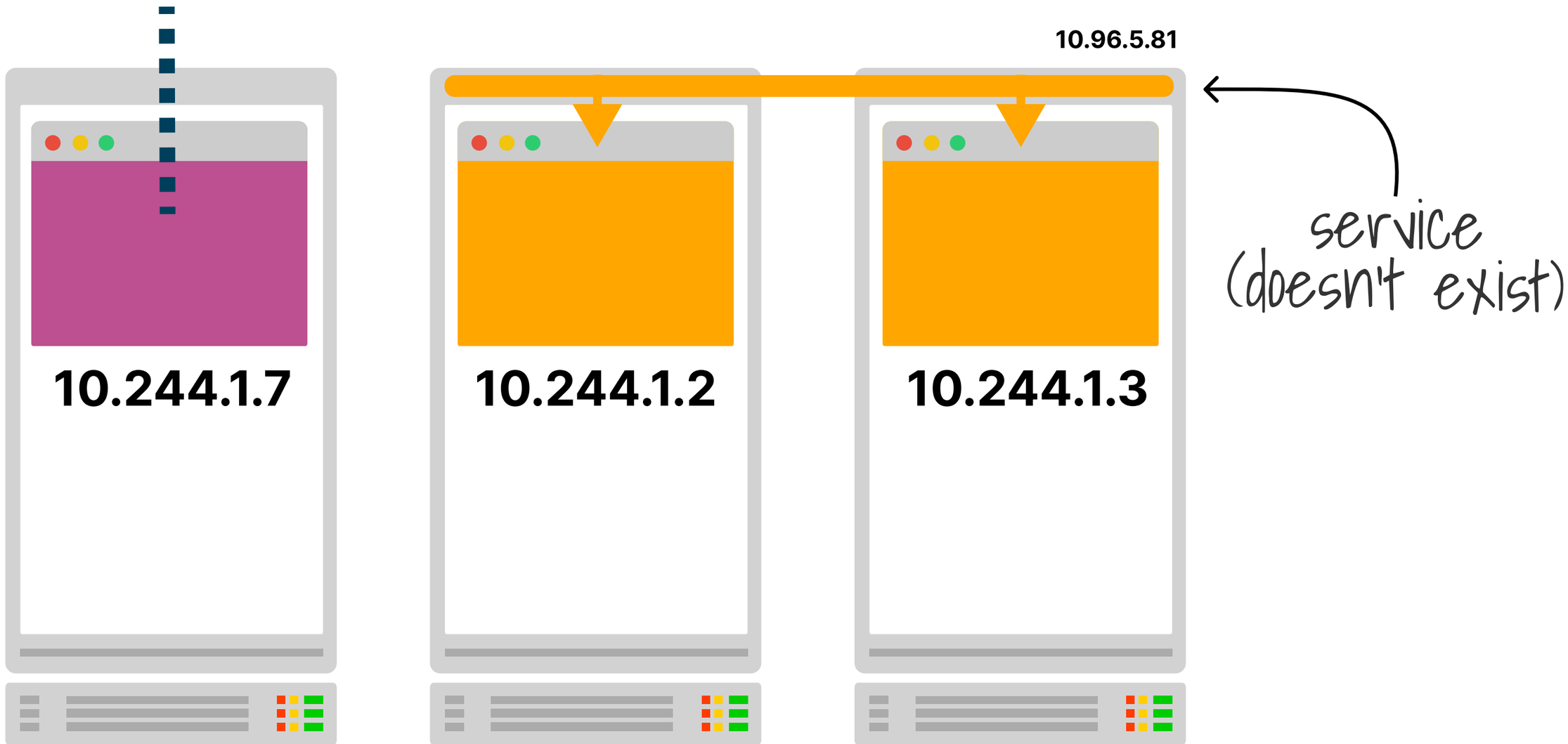
The kube-proxy component, running on each node, is responsible for programming these iptables rules.

FROM:

Pod (10.244.1.7)

TO:

Service (10.96.5.81)



When traffic reaches the node, it is intercepted by the iptables chains set up by kube-proxy.

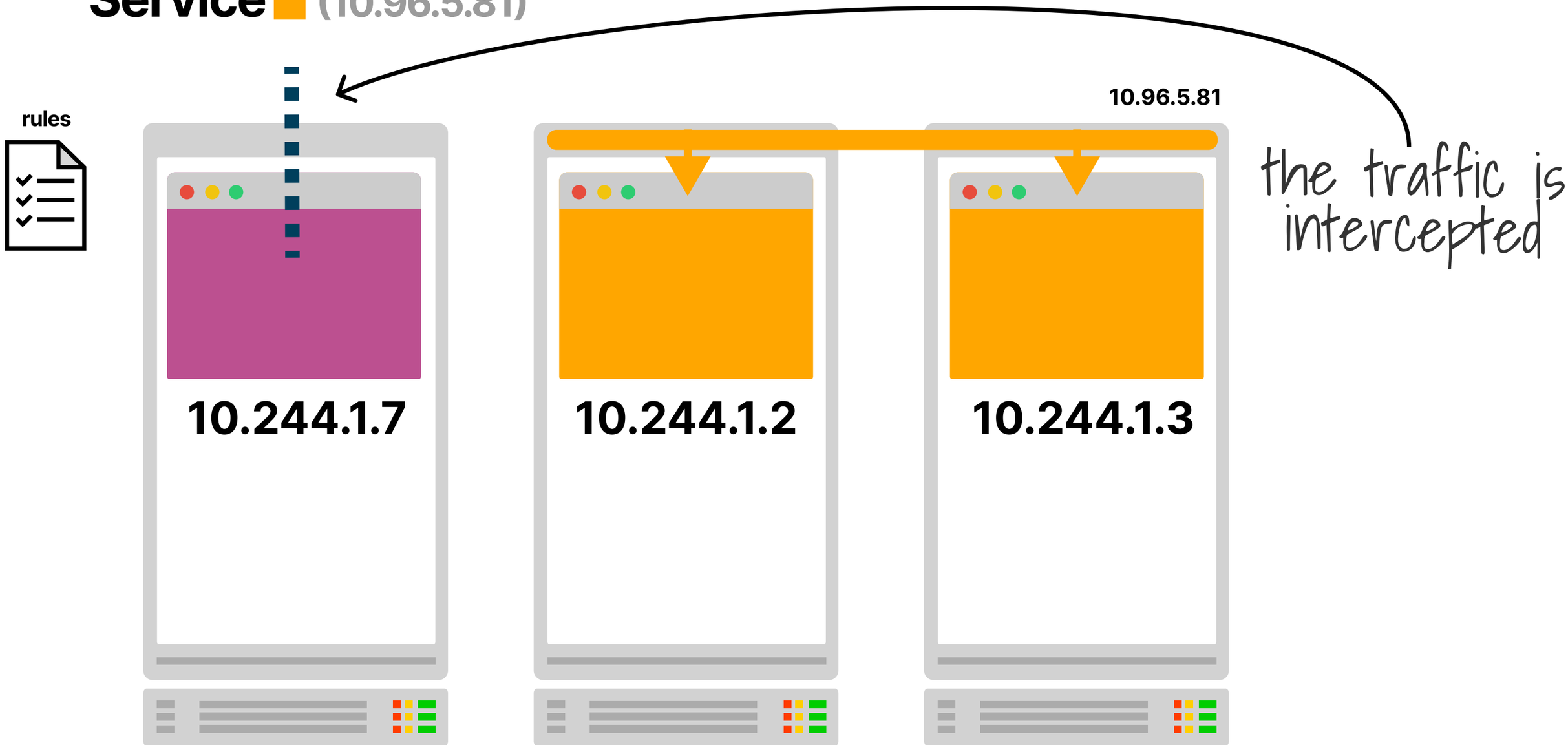
These chains are responsible for routing the traffic to the appropriate pods backing the Service.

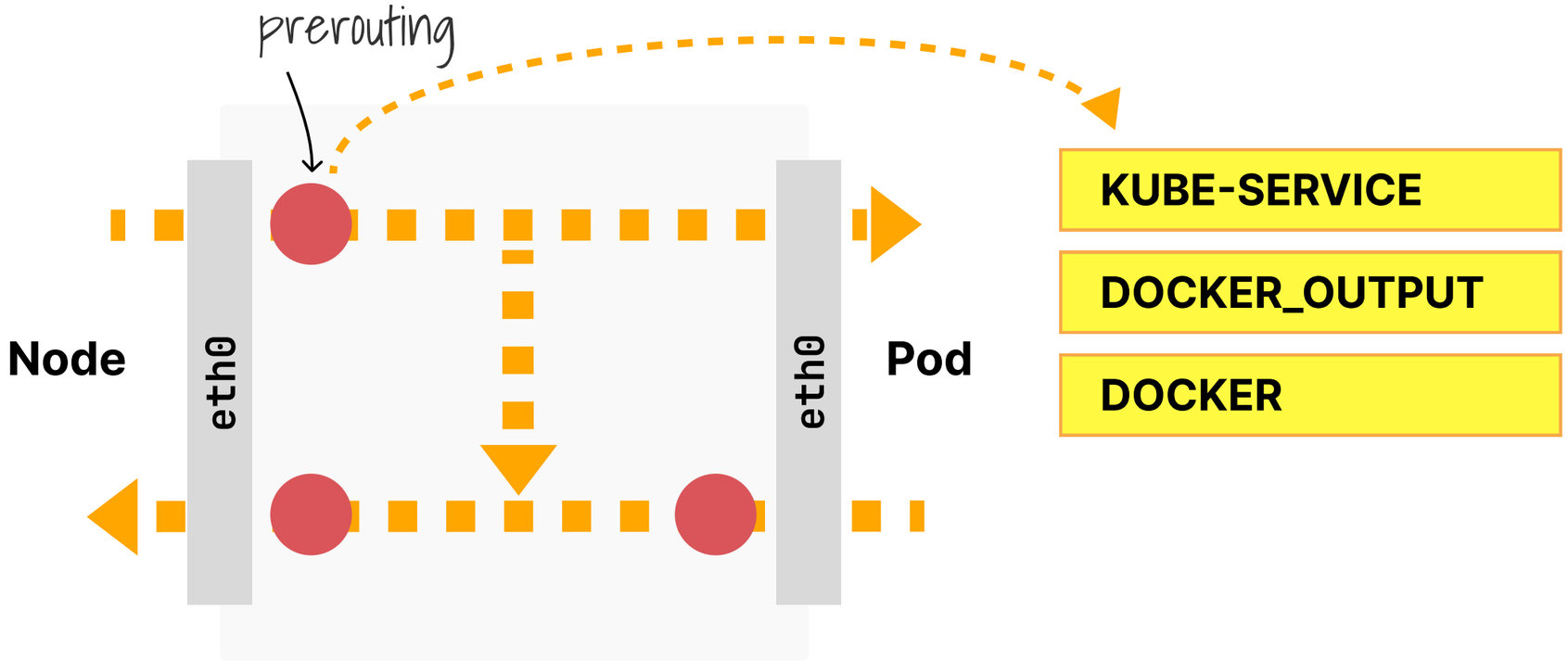
FROM:

Pod ■ (10.244.1.7)

TO:

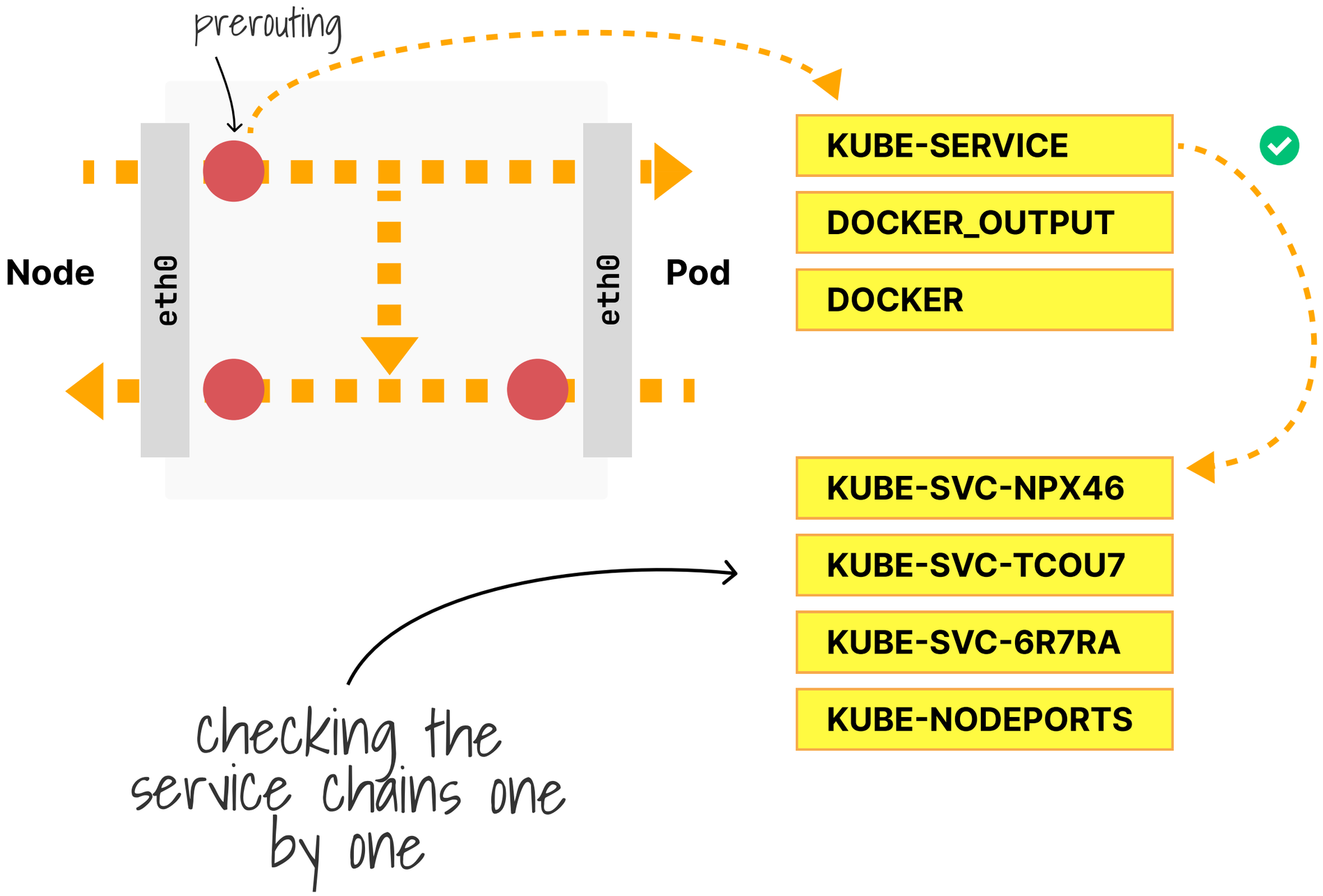
Service ■ (10.96.5.81)





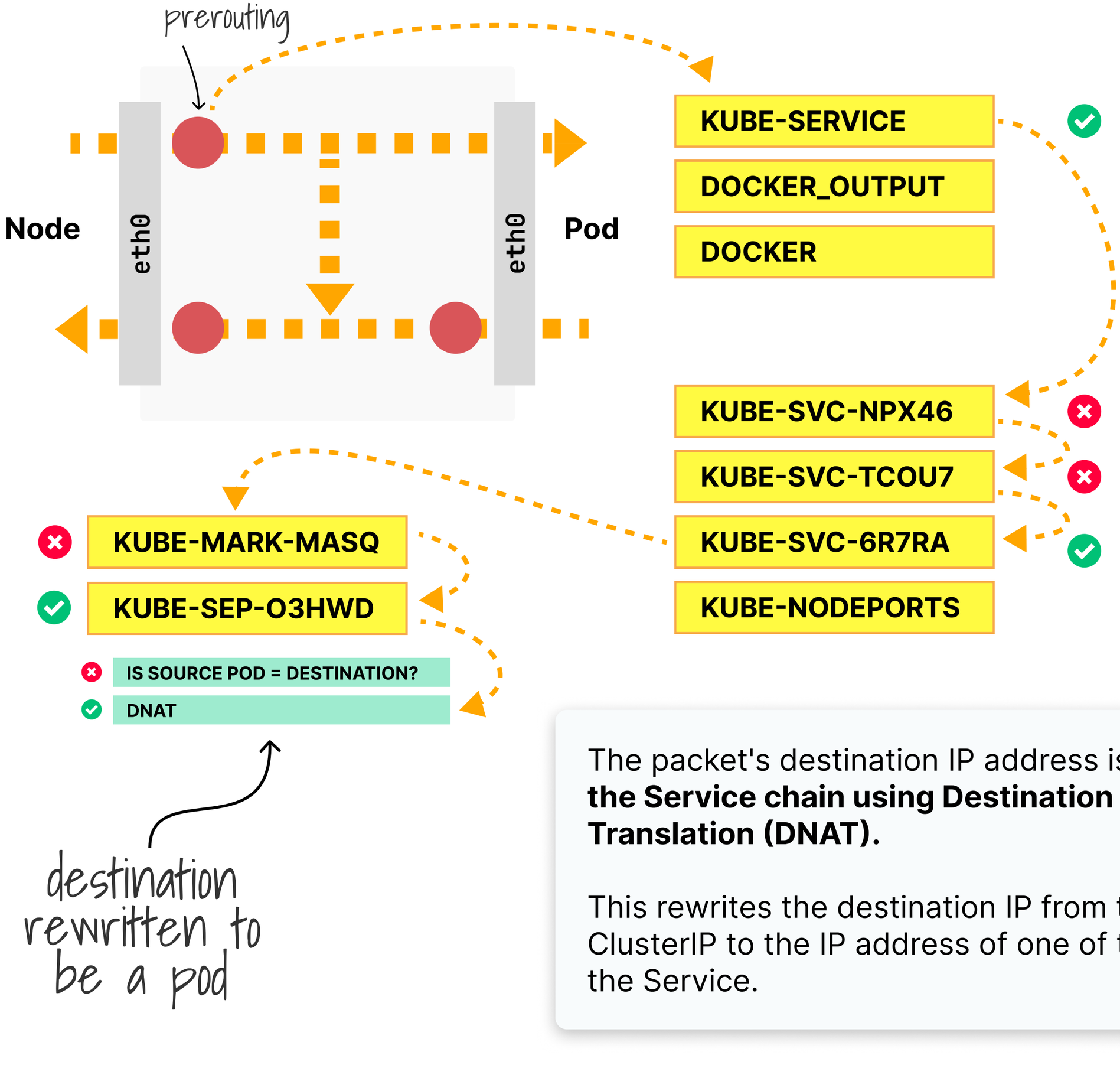
The first chain invoked is the `KUBE-SERVICES` chain, which contains rules for all Services in the cluster.

This chain is a collection of individual Service chains.



Each service has a corresponding Service chain.

The traffic is matched against these chains one by one, and if a match is found, the traffic is forwarded to that specific Service chain.

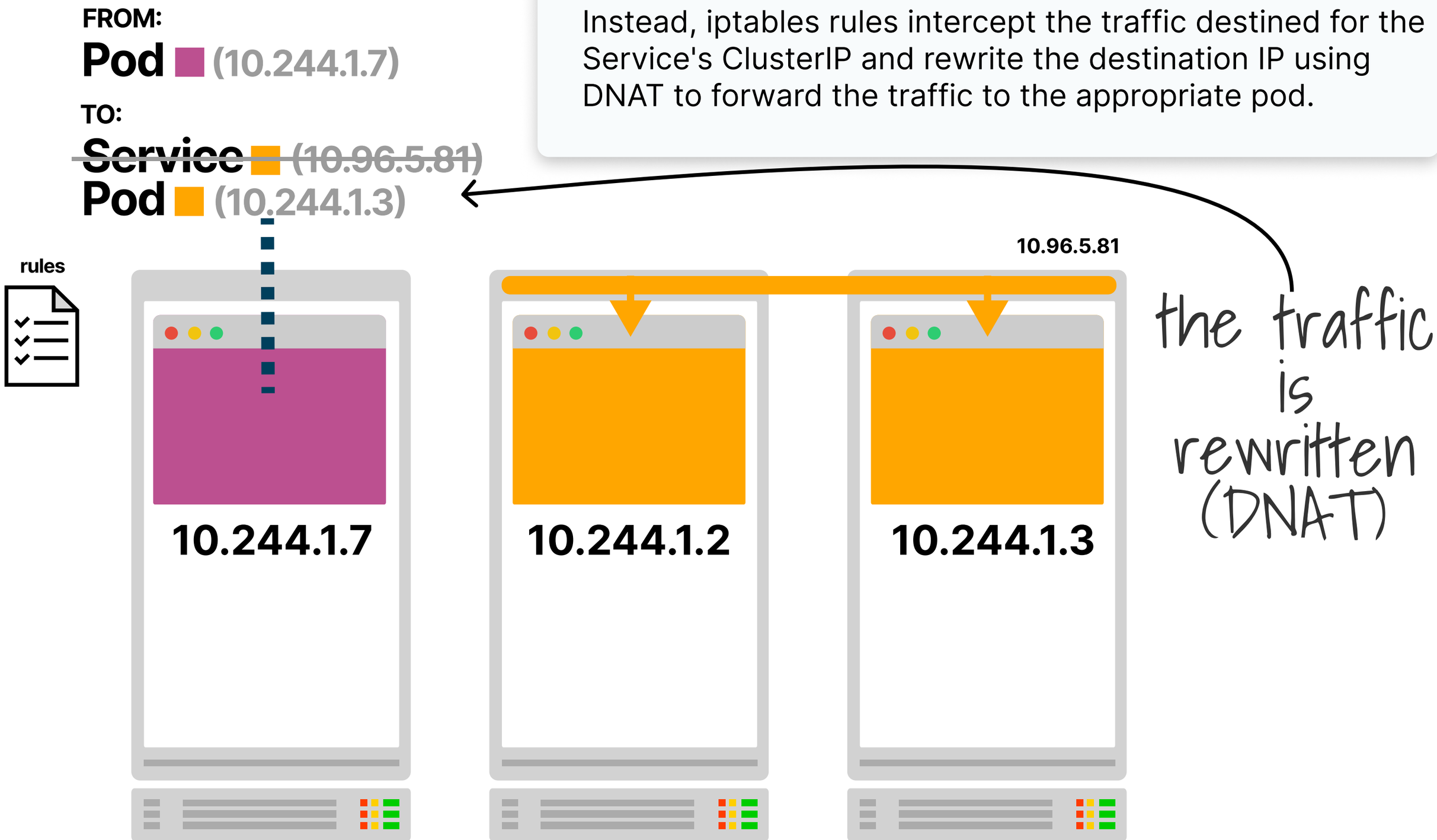


The packet's destination IP address is **rewritten inside the Service chain using Destination Network Address Translation (DNAT).**

This rewrites the destination IP from the Service's ClusterIP to the IP address of one of the pods backing the Service.

Kubernetes Services don't exist as actual processes or network interfaces.

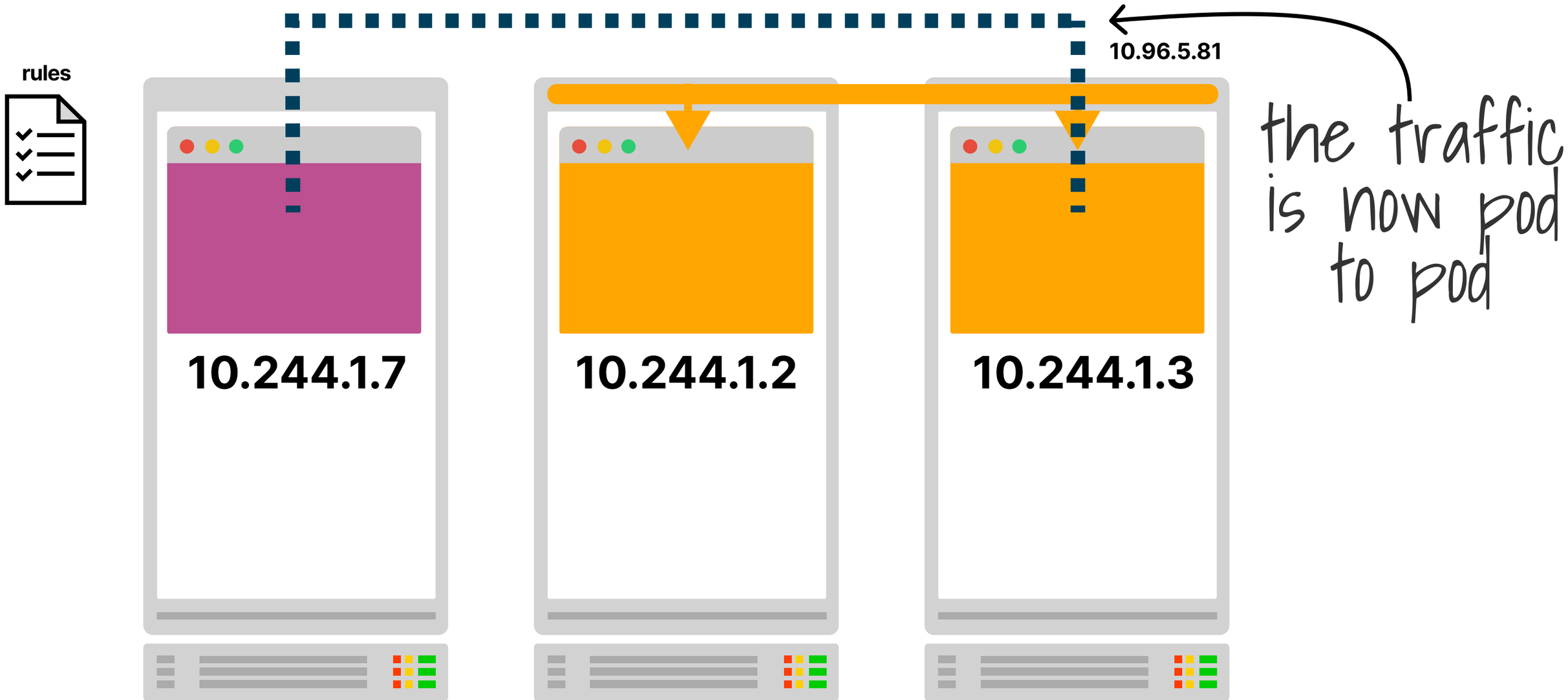
Instead, iptables rules intercept the traffic destined for the Service's ClusterIP and rewrite the destination IP using DNAT to forward the traffic to the appropriate pod.



Once the destination IP translation is complete, **the traffic flows directly from the node to the pod**, bypassing intermediate processes or load balancers.

FROM:
Pod (10.244.1.7)

TO:
Pod (10.244.1.3)



Kubernetes networking: servi

learnk8s.io/kubernetes-servi...

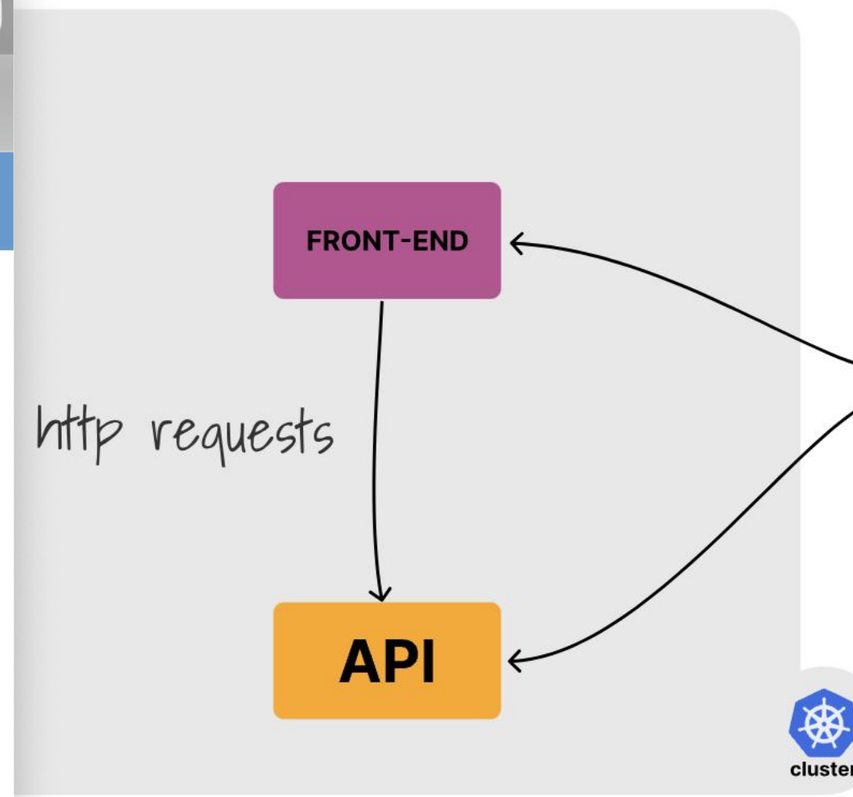
learnk8s

CORPORATE TRAINING PUBLIC WORKSHOPS BLOG CONTACT US

Gulcan Topcu

Kubernetes networking: service, kube-proxy, load balancing

OCTOBER 2024



front end calls the backend to display a job title and logs
essed the request.

deploy and expose those applications in Kubernetes.

ploying the Backend Pods

is what `backend-deployment.yaml` looks like.

ce that we will include `replicas: 1` to indicate that I wa
od.

learnk8s.io/kubernetes-services-and-load-balancing

TL;DR: This article explores Kubernetes networking, focusing on Services