

Tales Of The Kubernetes Ingress Networking: Deployment Patterns For External Load Balancers



How To Access The Slides?



- Slides (HTML): <https://containous.github.io/slides/webinar-cncf-jan-2020>
- Slides (PDF): <https://containous.github.io/slides/webinar-cncf-jan-2020/slides.pdf>
- Source on : <https://github.com/containous/slides/tree/webinar-cncf-jan-2020>

How To Use The Slides?

- **Browse the slides:** Use the arrows
 - Change chapter: Left/Right arrows
 - Next or previous slide: Top and bottom arrows
- **Overview of the slides:** keyboard's shortcut "o"
- **Speaker mode (and notes):** keyboard's shortcut "s"

Whoami

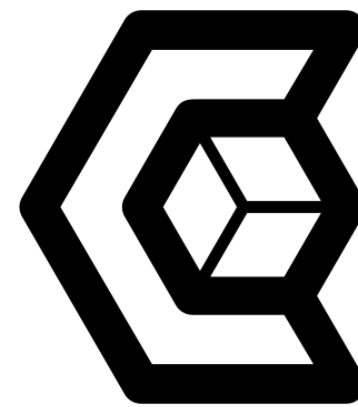
- Manuel Zapf:
 - Head of Product Open Source @ Containous
-  @mZapfDE
-  SantoDE



Containous

<https://containo.us>

- We Believe in Open Source
- We Deliver Traefik, Traefik Enterprise Edition and Maesh
- Commercial Support
- 30 people distributed, 90% tech

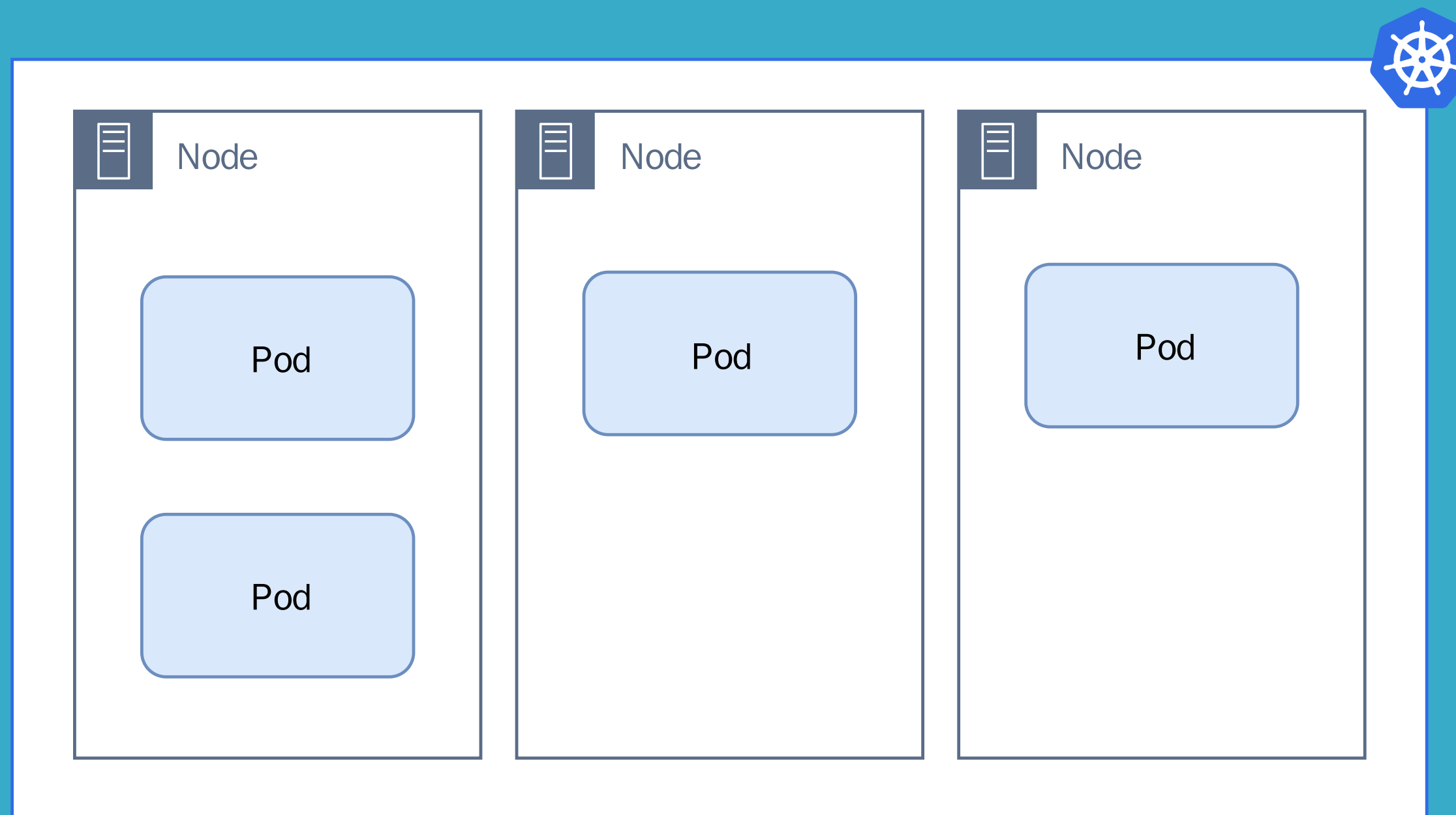


CONTAINOUS

Once Upon A Time

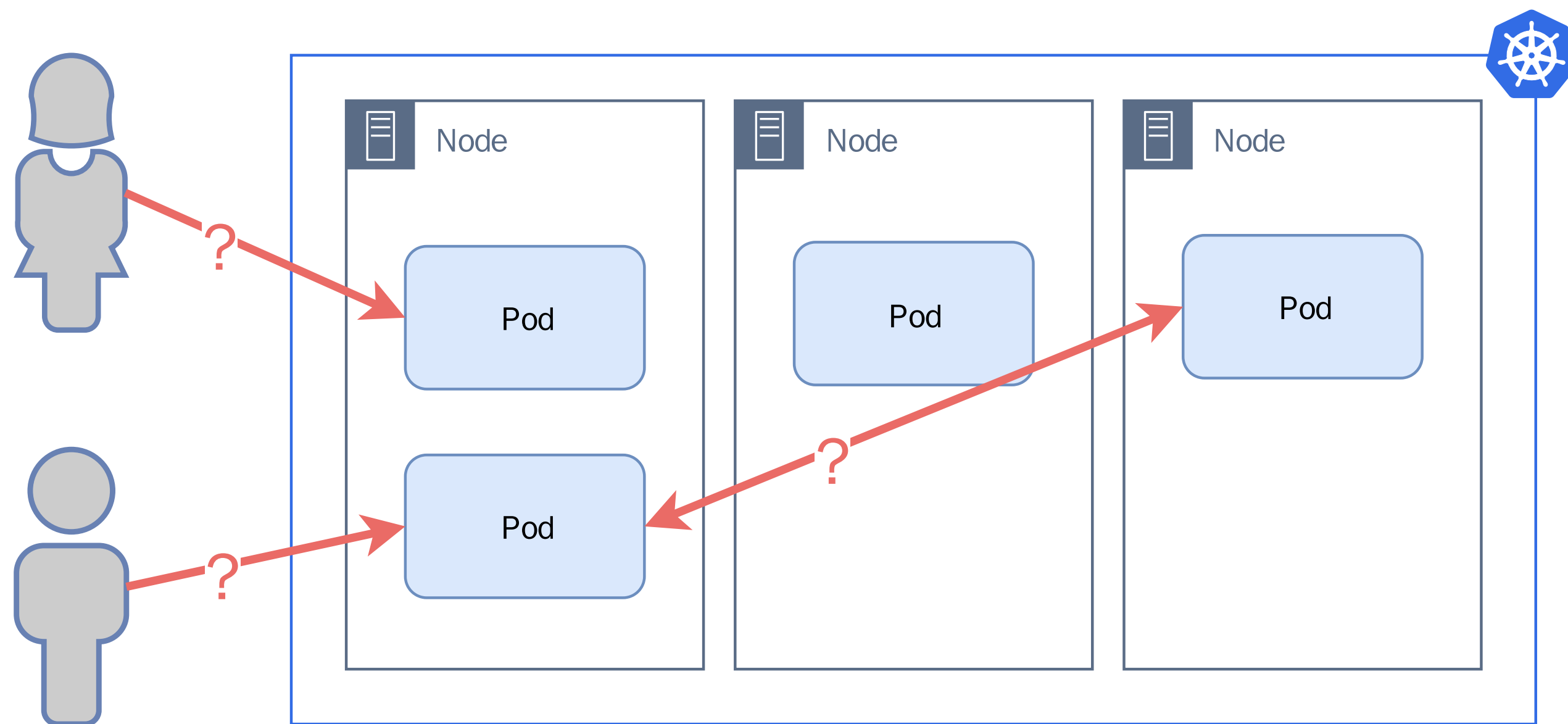
There was Kubernetes cluster.

This Cluster Had Nodes And Pods



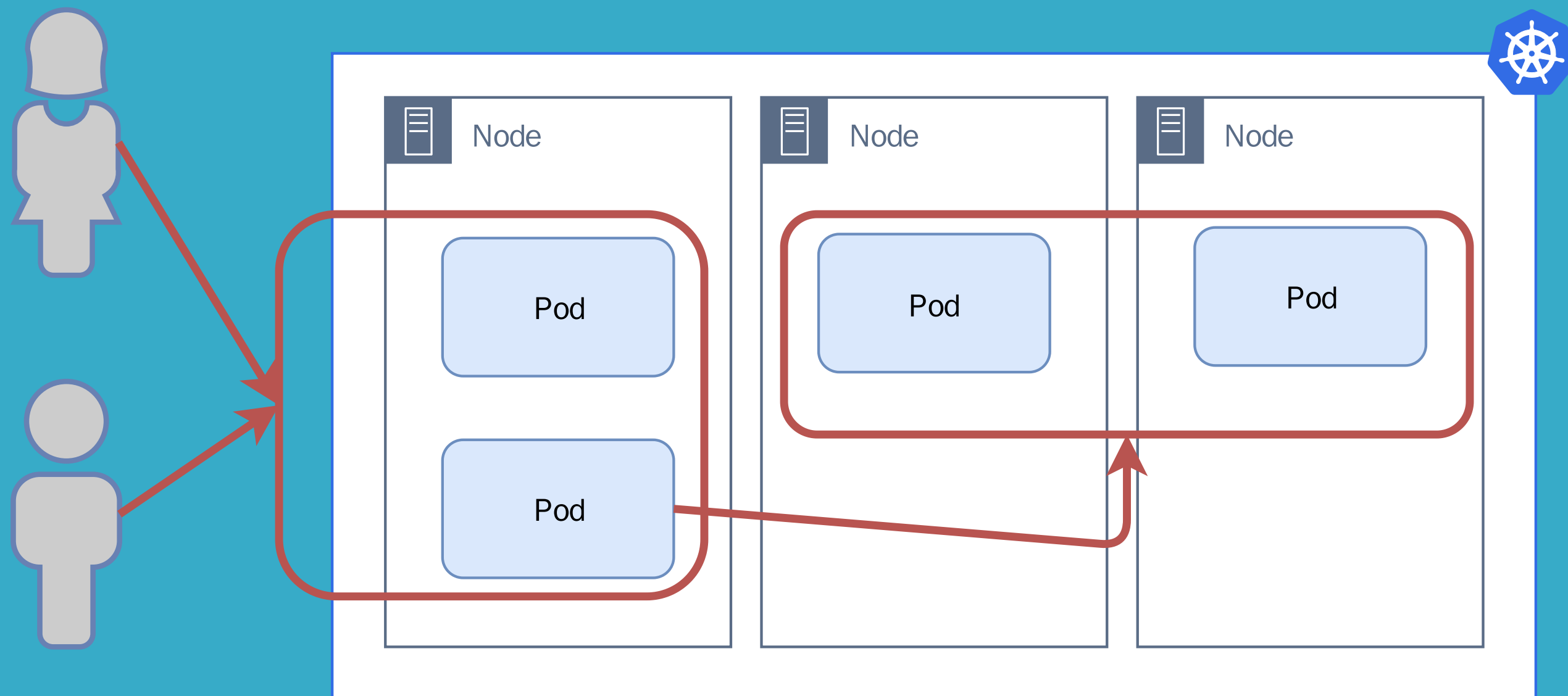
But Pods Had Private IPs

How to route traffic to these pods? And between pods on different nodes?



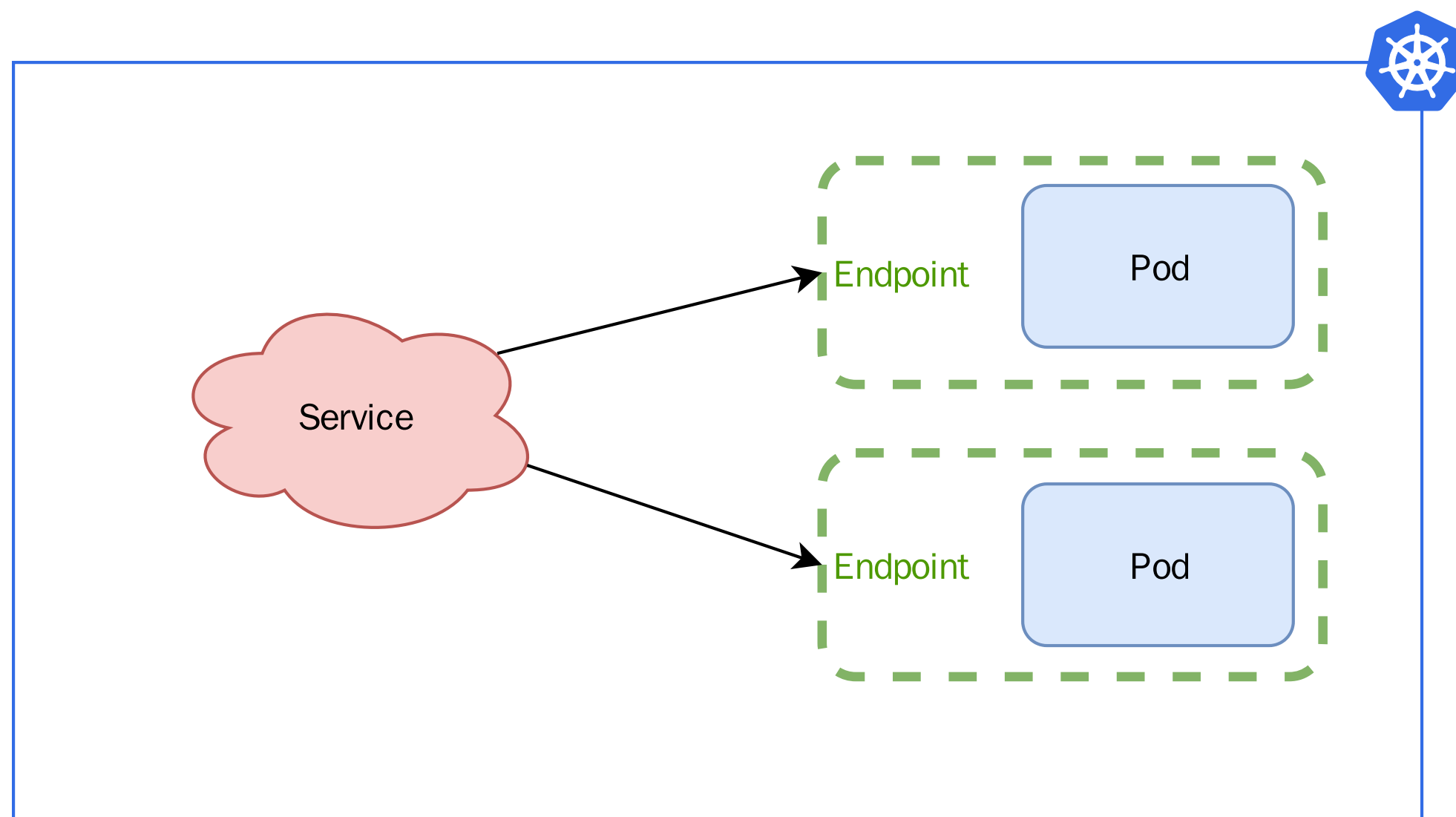
Services Came To The Rescue

Their goal: Expose Pods to allow incoming traffic



Services Are Load-Balancers

- Services have 1-N Endpoints
- EndPoints are determined by Kubernetes API



One exception: Services of types `ExternalName`

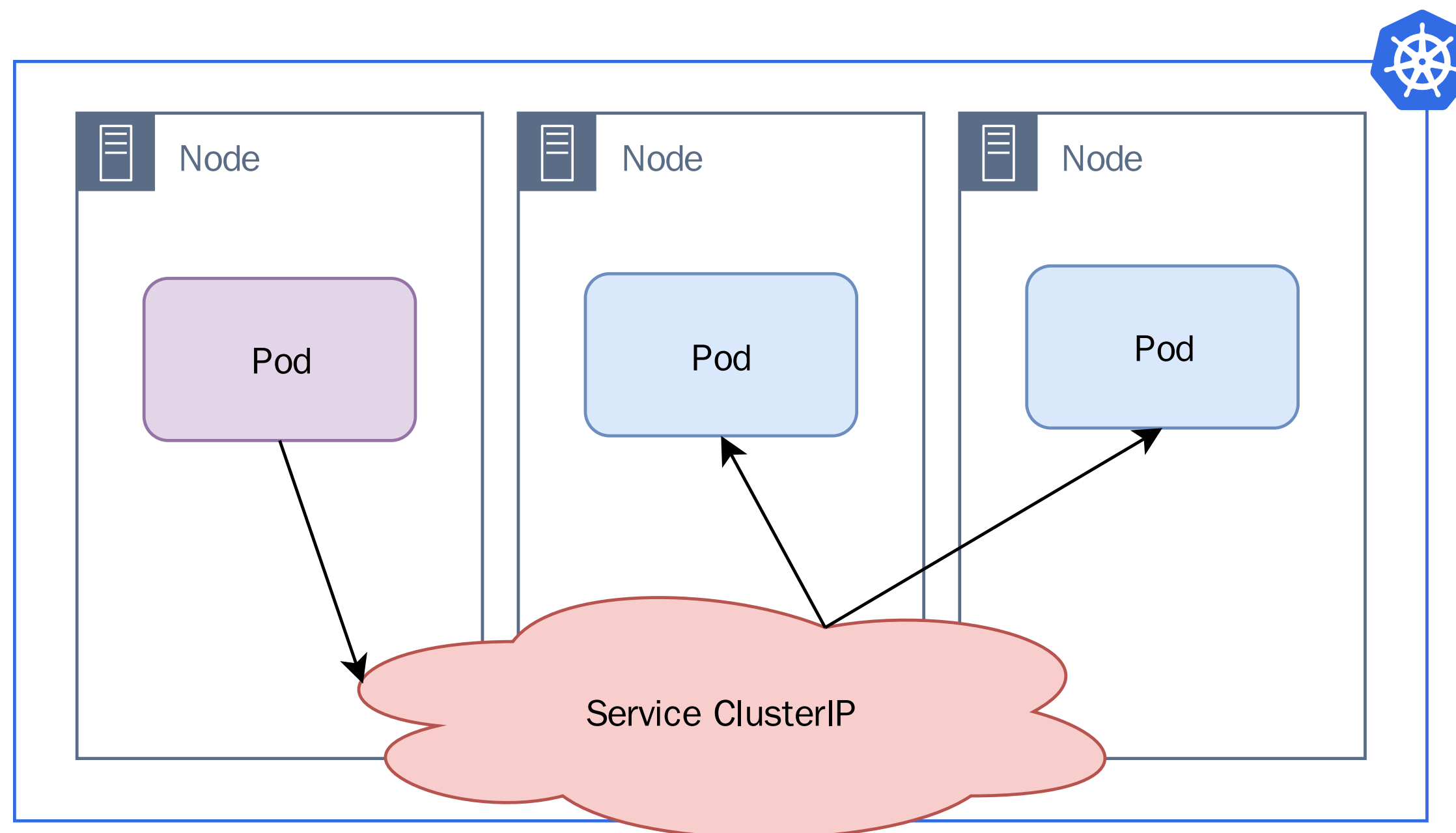
Different Kinds Of Services

for different communications use cases:

- From *inside*: type "ClusterIP" (default).
- From *outside*: types "NodePort" and "LoadBalancer".

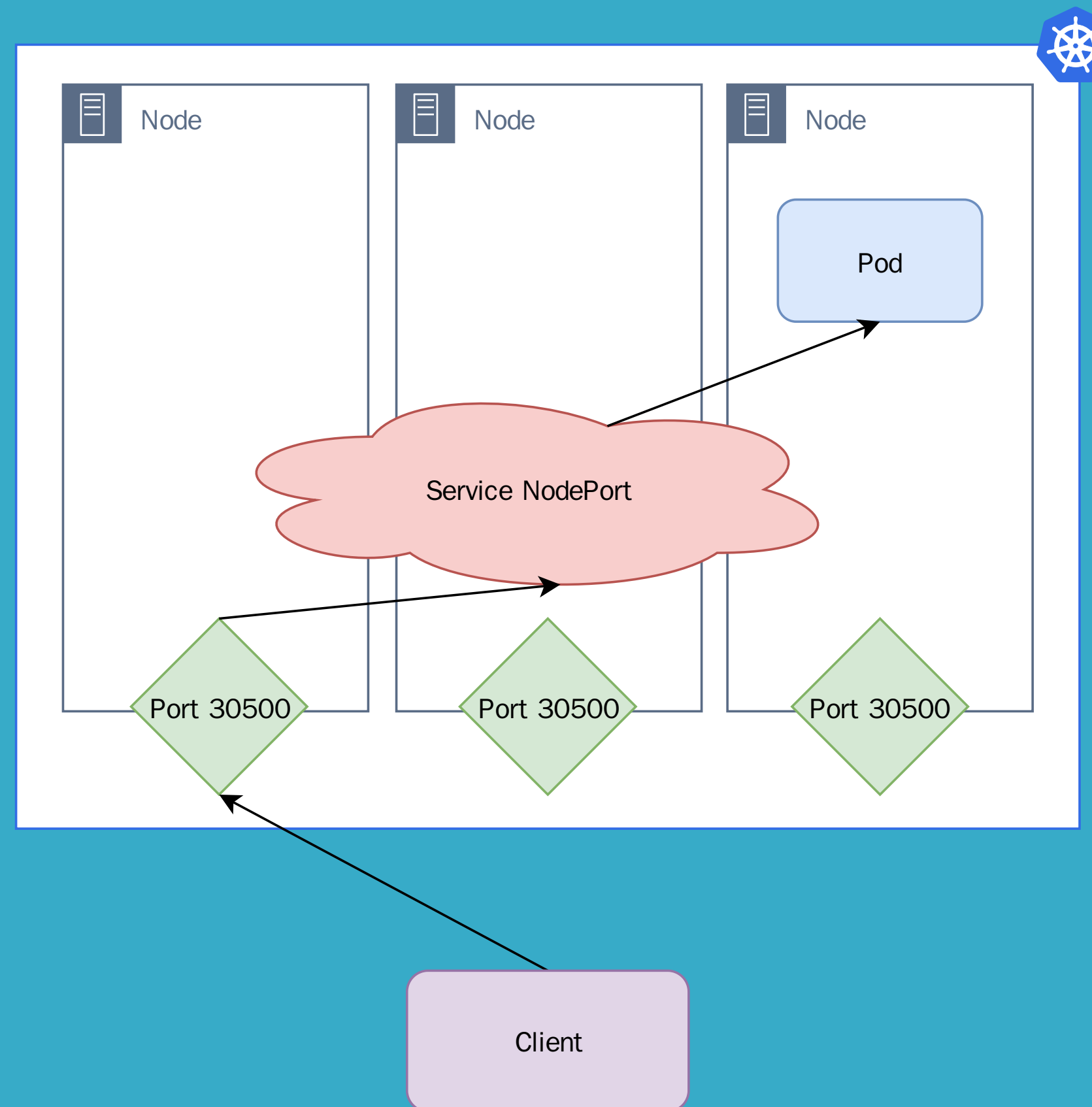
Services: ClusterIP

Virtual IP, private to the cluster, cluster)-wide (e.g. works from any node to any other node)



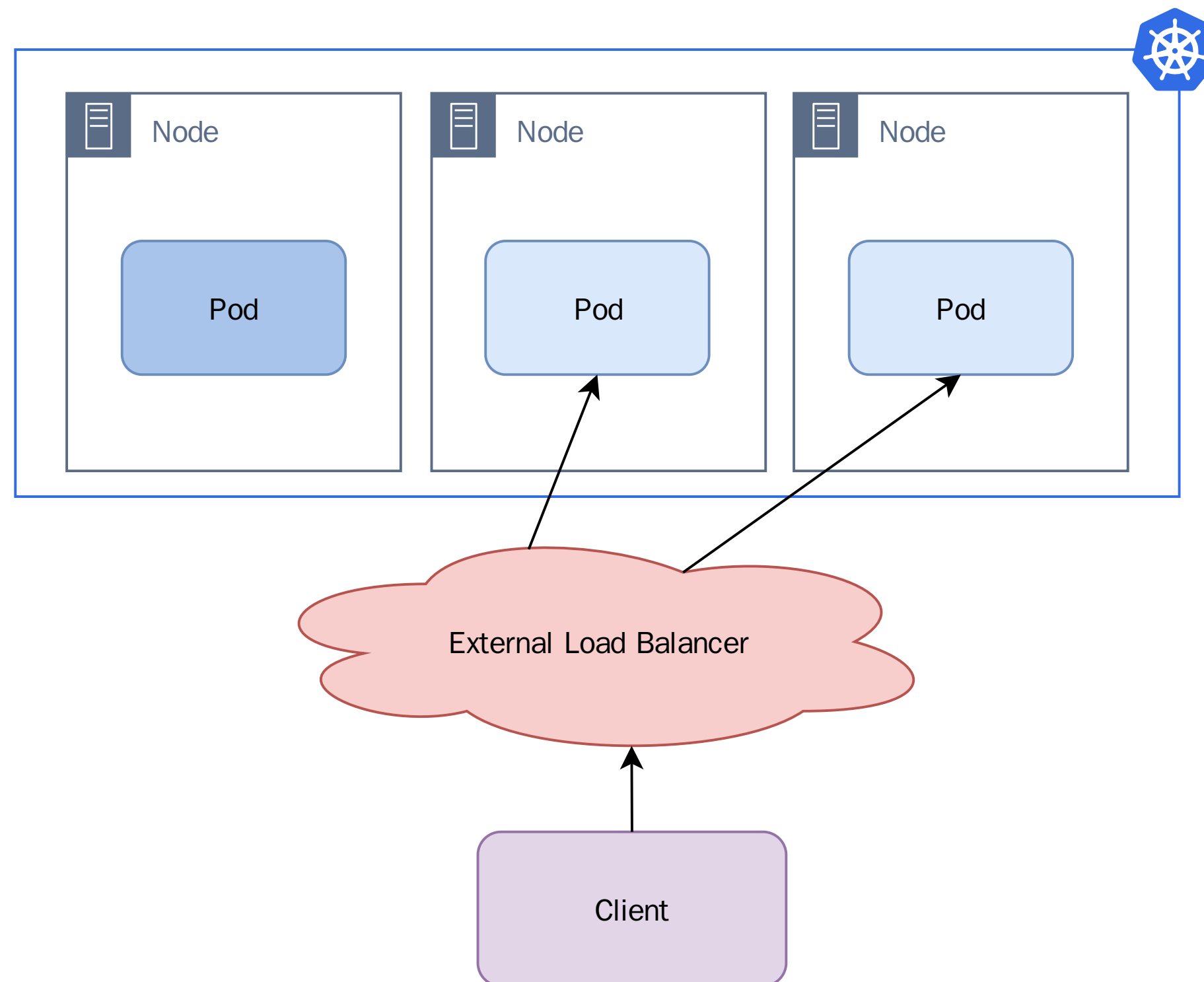
Services: NodePort

Uses public IPs and ports of the nodes, kind of "Routing grid"



Services: LoadBalancer

Same as NodePort, excepts it requires (and uses) an external Load Balancer.

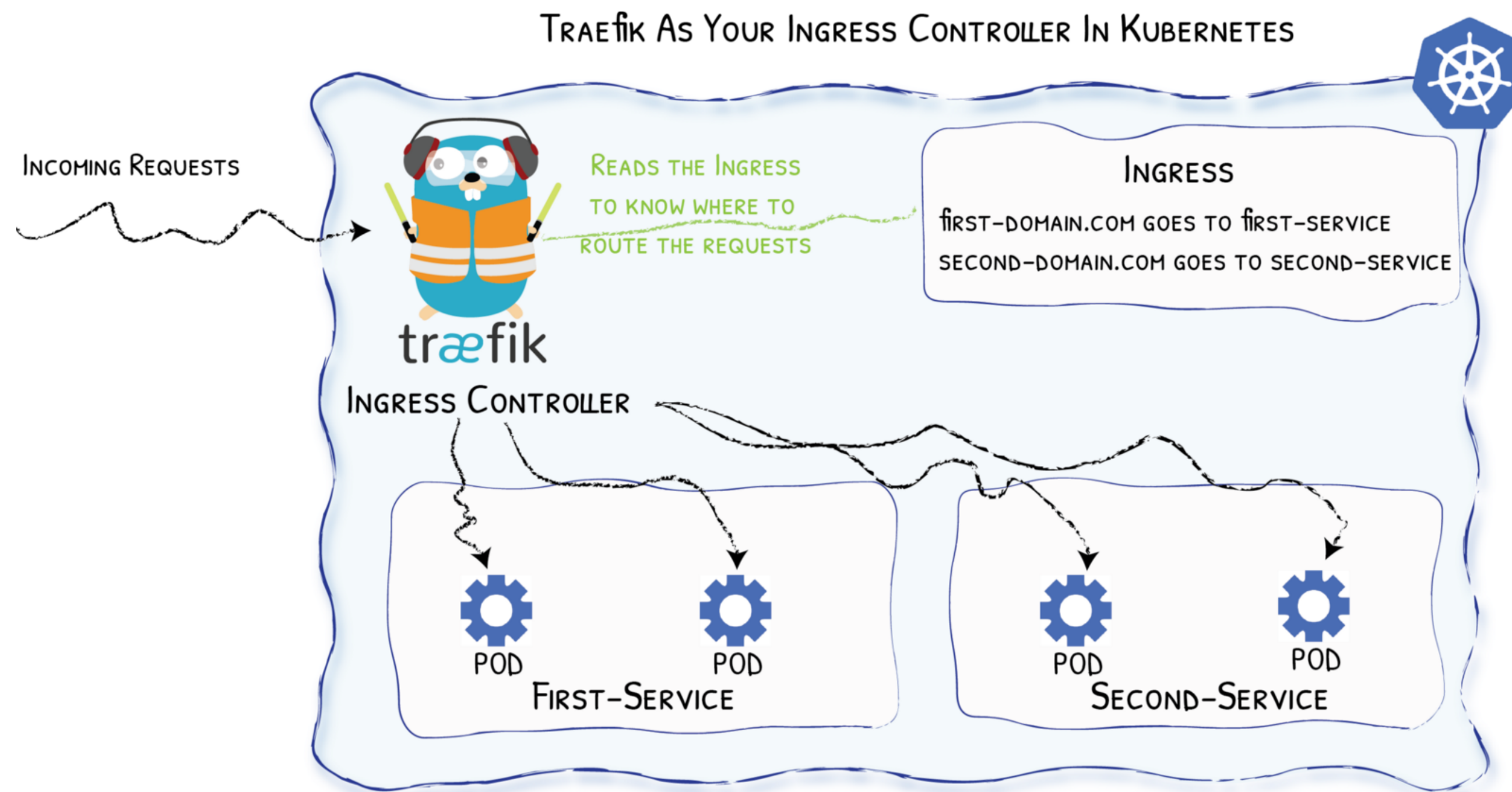


Services Are Not Enough

- Context: Exposes externally a bunch of applications
- Challenge: overhead of allocation for LBs. For each application:
 - One LB resource (either a machine or a dedicated appliance)
 - At least one public IP
 - DNS nightmare (think about the CNAMEs to create...)
 - No centralization of certificates, logs, etc.

And Then Came The Ingress

Example with Traefik as Ingress Controller:

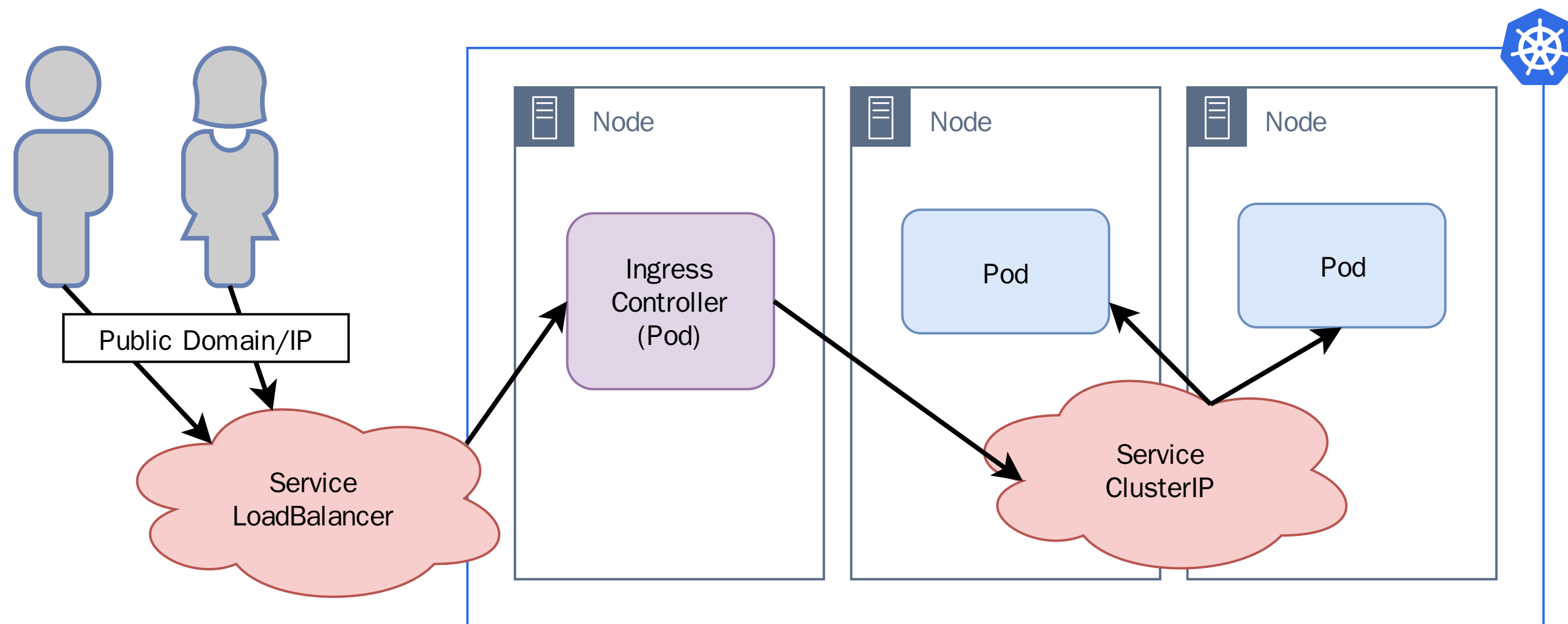


Notes About The Ingresses

Ingress Are Standard Kubernetes Applications

- Deployed as Pods (Deployment or as DaemonSet)
- Exposed with a Service:
 - You still need access from the outside
 - But only one service to deal with (ideally)

Ingress Have Services Too



Why Should I Care?

- Simplified Setup:
 - Single entriypoint, less configuration, better measures
 - Less resources used
 - Separation of concerns: differents algorithms for load balancing, etc.

Why Challenges Does It Make?

- Designed for (simple) HTTP/HTTPS cases
 - TCP/UDP can be used, but are not first-class citizens
 - "Virtual Host First" centric
- Feels like you must carefully select your (only) Ingress Controller

So What?

- Kubernetes gives you freedom:
 - You can use multiple Ingress Controllers!
- Kubernetes gives you choices:
 - So much deployment patterns that you can do almost anything

External Load Balancers

Did You Just Say "External"?

- Outside the "Borders" of Kubernetes:
 - Depends on your "platform" (as in infrastructure/cloud)
- Still Managed by Kubernetes (Automation)
 - Requires "plugins" (operators/modules) per Load Balancer provider
 - No API or no Kubernetes support: requires switching to NodePort

Tell Me Your Kubernetes Distribution

...and I'll tell you which LB to use...

Cloud Managed Kubernetes

- Cloud providers provides their own external LBs
 - Fully Automated Management with APIs
 - Great UX due to the integration: works out of the box
 - Benefits from cloud provider HA and performances
- But:
 - You have to pay for this :)
 - Configuration is cloud-specific (using annotations)
 - Relies on LB implementation limits

Bare-Metal Kubernetes

Aka. "Run it on your boxes"

- Best approach: **Metal LB**, a Load Balancer implementation for Kubernetes, hosted **inside** Kubernetes
 - Uses all Kubernetes primitives (HA, deployment, etc.)
 - Allows Layer 2 routing as well as BGP
 - But... still not considered production ready
- Otherwise: external static (or legacy) LB
 - Requires switching to `NodePort` Service

Cloud "Semi-Managed" Kubernetes

- Depends on the compute provider: cloud or bare-metal
- You need a tool for managing clusters: kubeadm, kops, etc.
 - Most of these tools already manage LB if the provider does.

Source IP On The Kingdom Of Kubernetes

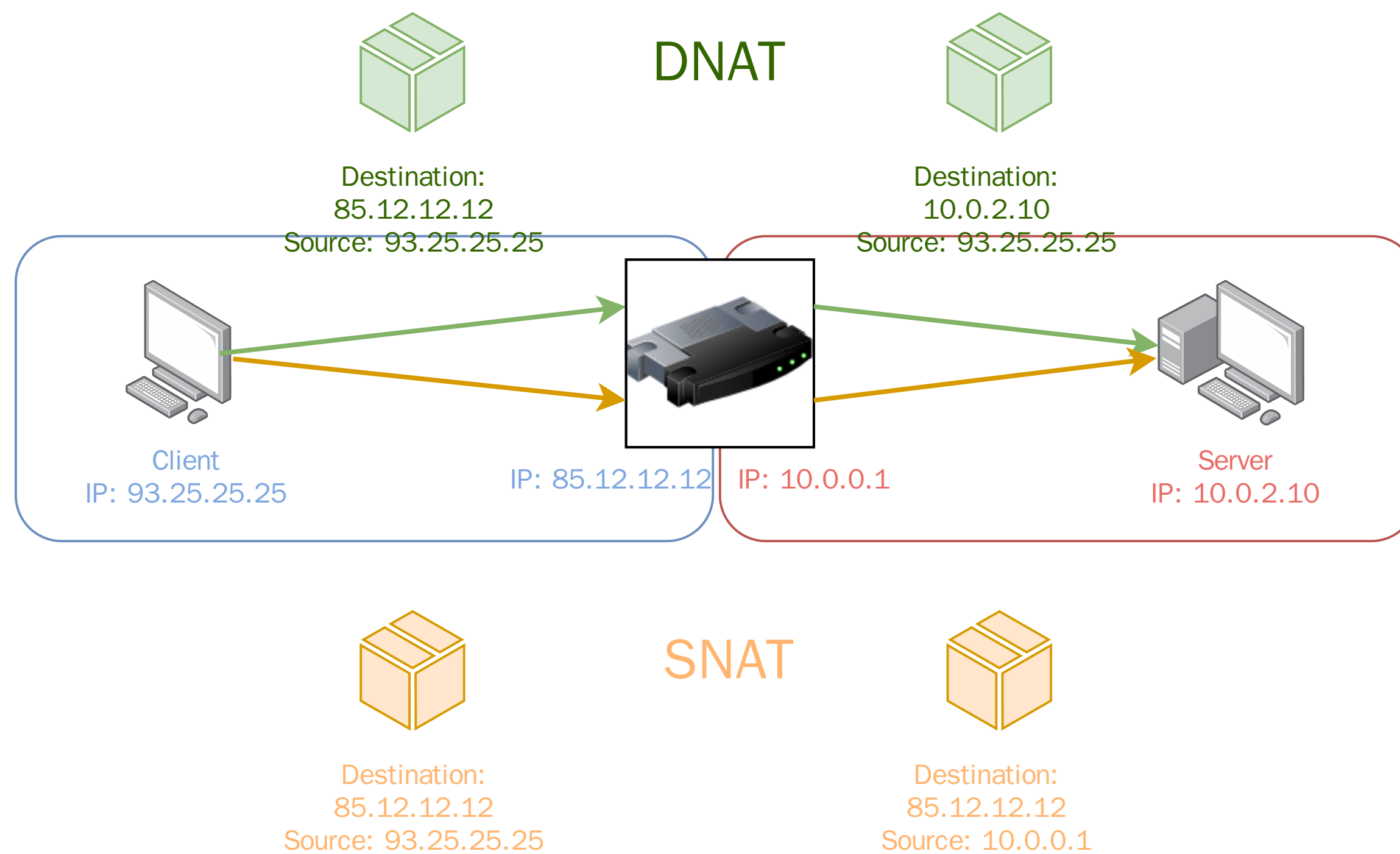
Business Case: Source IP

As a business manager, I need my system to know the IP of the emitters of the requests to track usage, write access logs for legal reasons and limit traffic in some cases.

NAT/DNAT/SNAT

- **NAT** stands for "Network Address Translation"
 - IPv4 world: Routers "masquerades" IPs, to allow routing from different network
- **DNAT** stands for "Destination NAT"
 - Masquerade of the destination IP with the internal pod IP
- **SNAT** stands for "Source NAT"
 - Masquerade of the *source* IP with the router's IP

NAT/DNAT/SNAT



Preserve Source IP

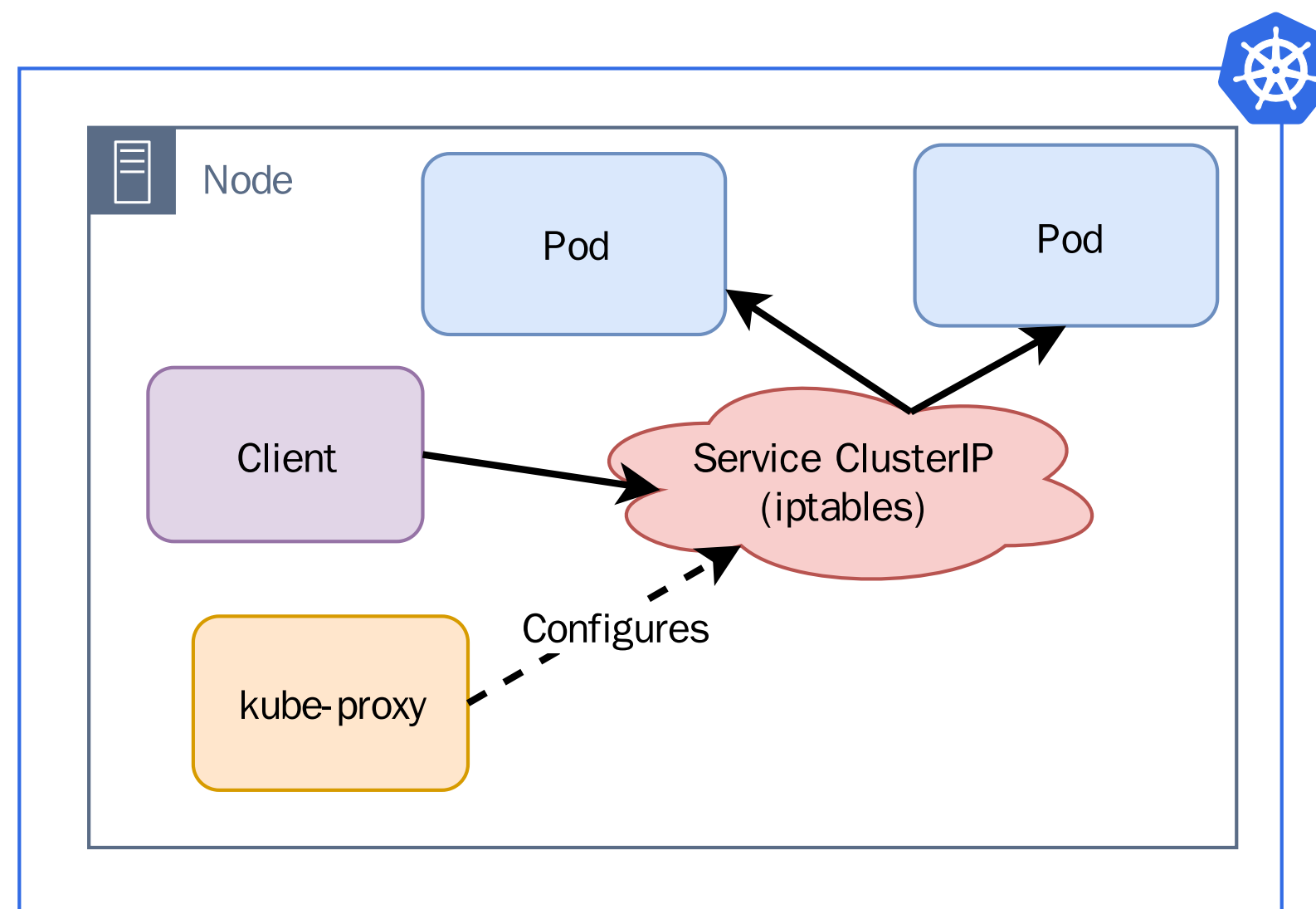
- Rule: We do NOT want SNAT to happen
- Challenge: many intermediate components can interfere and SNAT the packets in our back!

Inside Kubernetes: Kube-Proxy

- `kube-proxy` is a Kubernetes component, running on each worker node
- Role: manage the virtual IPs used for Services
- Challenge with Source IP: `kube-proxy` might SNAT requests
- SNAT by `kube-proxy` depends on the Service:
 - Let's do a tour of Services Types!

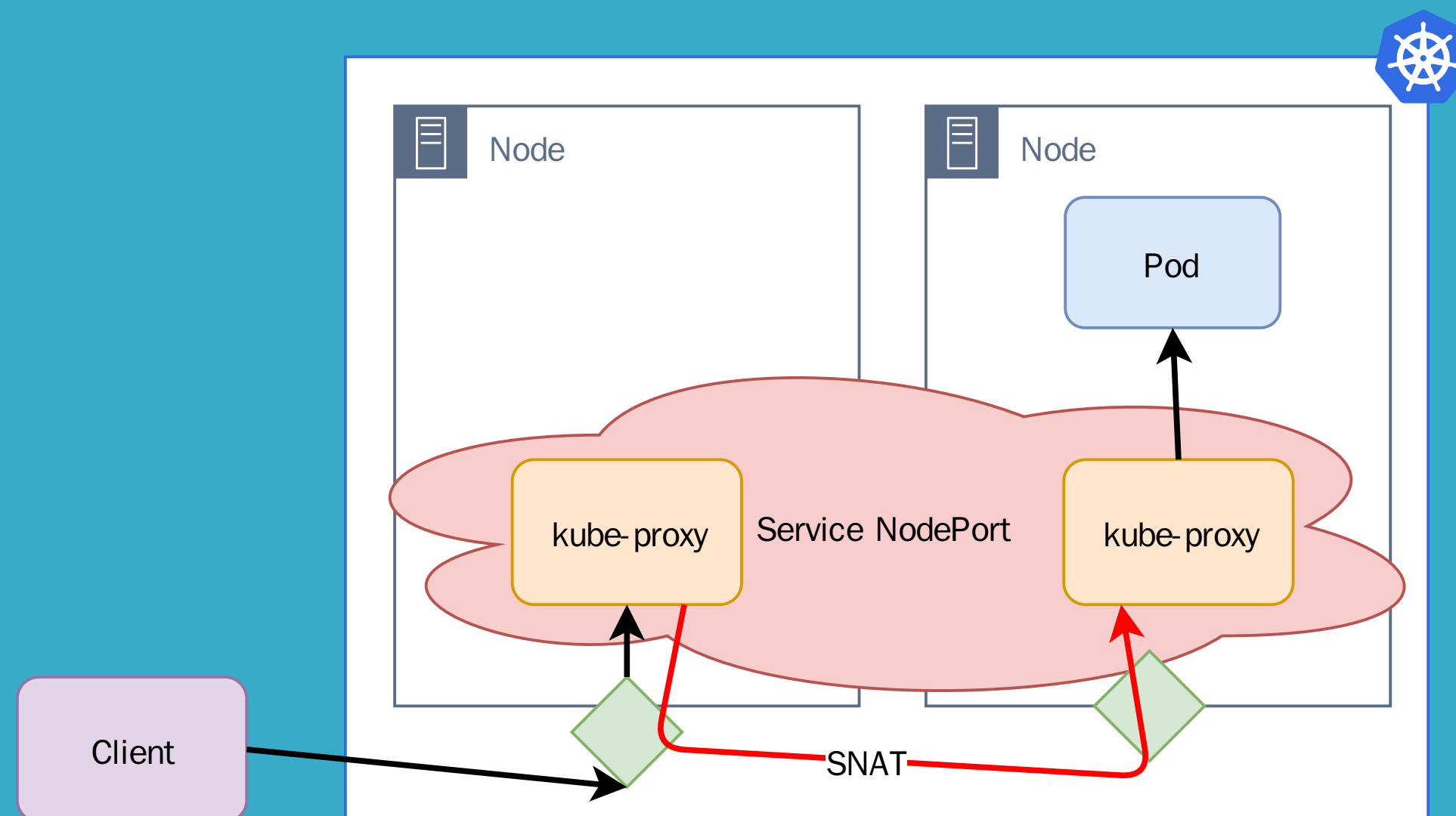
Source IP With Service ClusterIP

- When `kube-proxy` is in "iptables" mode: no SNAT ✓
 - This is the default mode
 - No intermediate component



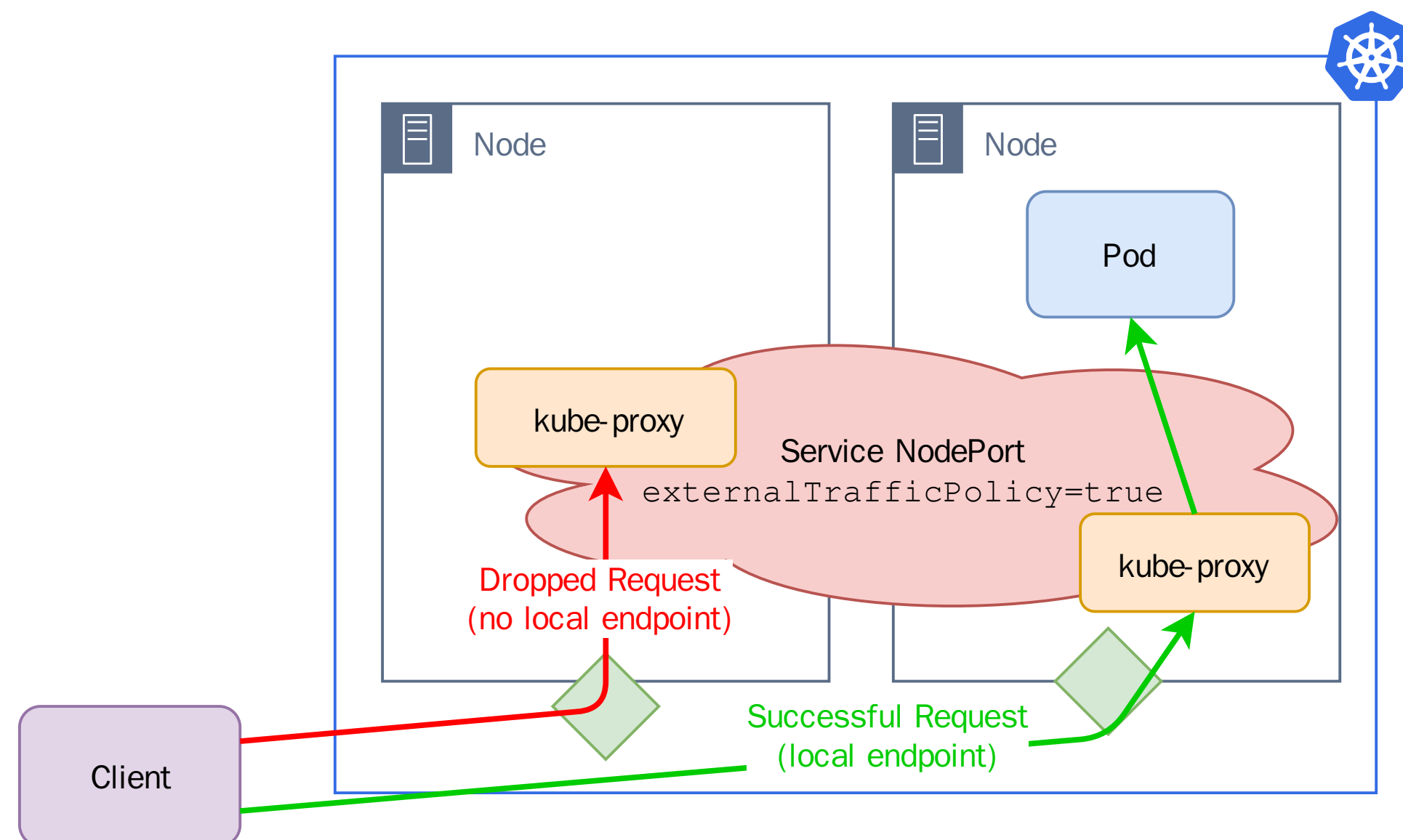
Source IP With Service NodePort (Default)

- SNAT is done ✕ (routing to the node where pod is):
 - First node to node routing through nodes network
 - Then node to pod routing through pod network



Source IP With Service NodePort (Local Endpoint)


- No SNAT ✓ with `externalTrafficPolicy` set to `Local`
- Downside: Dropped request if no pod on receiving node



Source IP With Service LoadBalancer (Default)

- Default: SNAT is done ✕, same as NodePort
 - External Load Balancer can route to any node
 - If no local endpoint: Node to node routing with SNAT

Source IP With Service LoadBalancer (Local Endpoint)

- However, No SNAT ✓ for load balancers implementing Local externalTrafficPolicy:
 - GKE/GCE LB, Amazon NLB, etc.
 -  Nodes without local endpoints are removed from the LB by failing healthchecks
 - ☐☐Pros: no dropped request from client view, but nodes always ready
 - ☐☐Cons: relies on healthcheck timeouts

Alternatives When SNAT Happen

- Sometimes, SNAT is mandatory
 - External LB
 - Network Constraint
 - Ingress Controller in the middle
- "Network is based on layers" - let's use another layer:
 - If using HTTP, retrieve the Source IP from headers
 - If using TCP/UDP, use the "Proxy Protocol"
 - Or use distributed logging and tracing


HTTP Protocol Headers

- X-Forwarded-From holds a comma-separated list of all the source IPs SNAT during all network hops.
 - ✓ if you have an External LoadBalancer or an Ingress Controller supporting this header.
 - ⚠ Not standard (header starting with x-) so not all HTTP appliance might support it.
 - Upcoming Official HTTP Header Forwarded

Proxy Protocol

- Introduced by HAProxy
- Happens at Layer 4 (Transport) for TCP/UDP
- Goal: "chain proxies / reverse-proxies without losing the client information"
- Supported by a lot of appliances in 2019: AWS ELB, Traefik, Apache, Nginx, Varnish, etc.
- Use Case: when SNAT happen AND not way to use HTTP. H

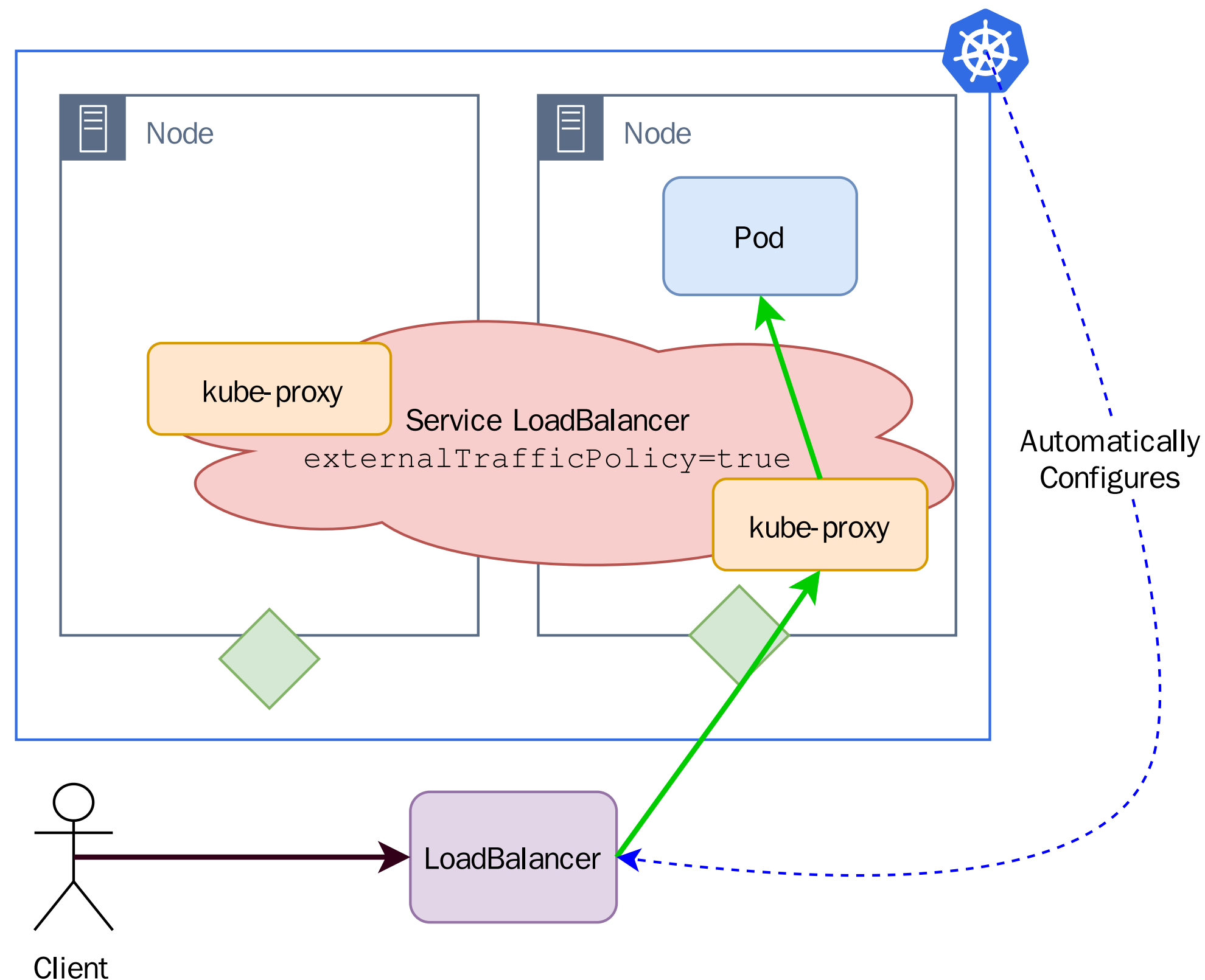
Distributing Logging And Tracing

-  *Idea:*
 - Collect the source IP as soon as possible in distributed logging
 - Use distributed tracing to track the request in the system
- ☐☐Pros: no more complex network setups, distributed logging and tracing stacks are already on your Kubernetes cluster (or will soon be)
- ☐☐Cons: relies on the distributed logging/tracing stacks

Use Cases

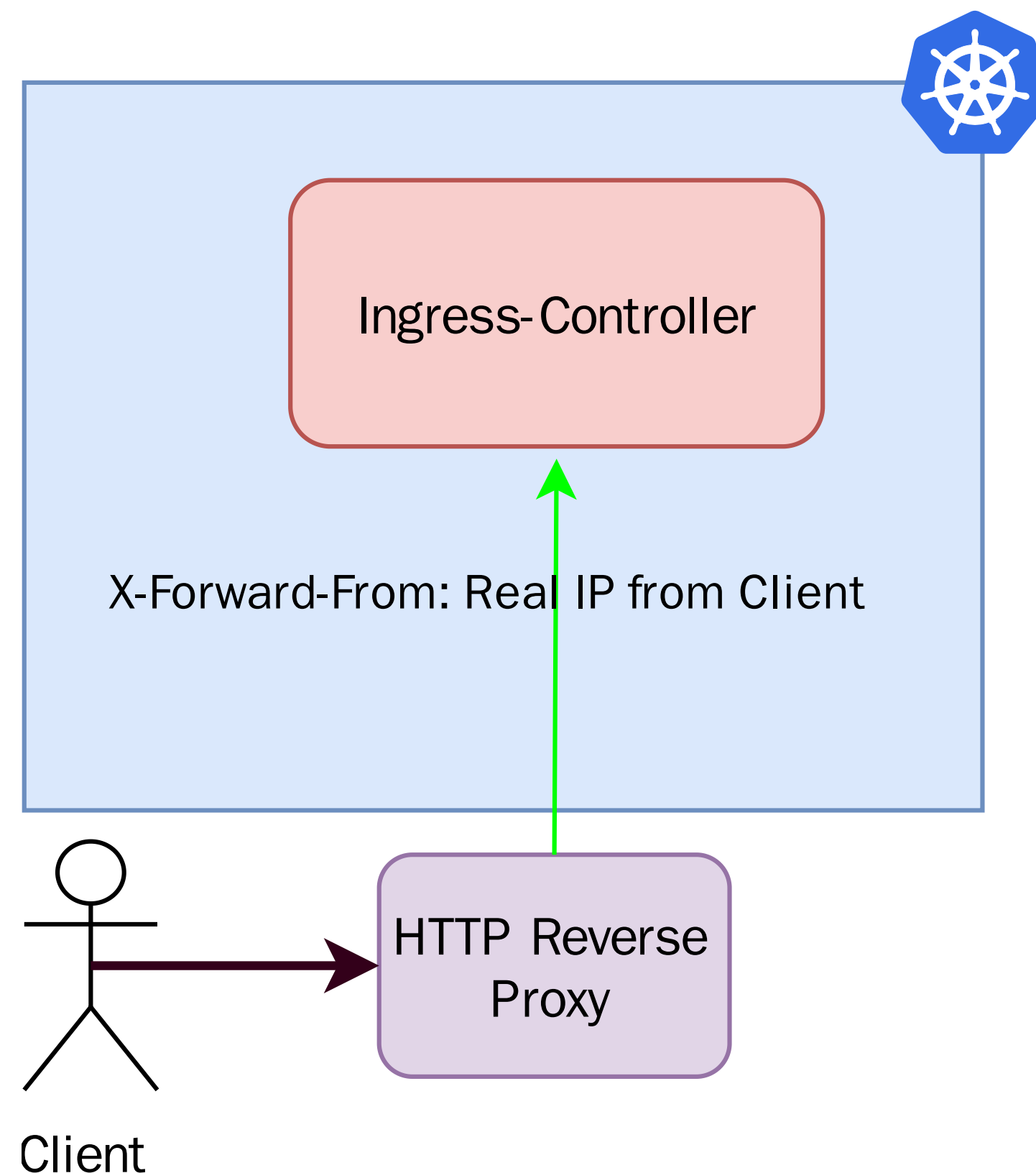
External Load Balancer With Traffic Policy

- ☑ Pros: full automation
- 👎 Cons: depends on actual LB implementation



Capturing Source IP From HTTP Headers

- ☑ Pros: Simplified Setup
- 👎 Cons: Only works with HTTP



Sources

- <https://kubernetes.io/docs/tutorials/services/source-ip/>
- https://en.wikipedia.org/wiki/Network_address_translation
- <https://www.asykim.com/blog/deep-dive-into-kubernetes-external-traffic-policies>

Read More



<https://info.containo.us/request-white-paper-routing-in-the-cloud>

Thank You!

 @mZapfDE

 SantoDE



- Slides (HTML): <https://containous.github.io/slides/webinar-cncf-jan-2020>
- Slides (PDF): <https://containous.github.io/slides/webinar-cncf-jan-2020/slides.pdf>
- Source on : <https://github.com/containous/slides/tree/webinar-cncf-jan-2020>