Tales of the Kubernetes Ingress Networking: Einsatzmuster für externe Load Balancer

4. November 2020

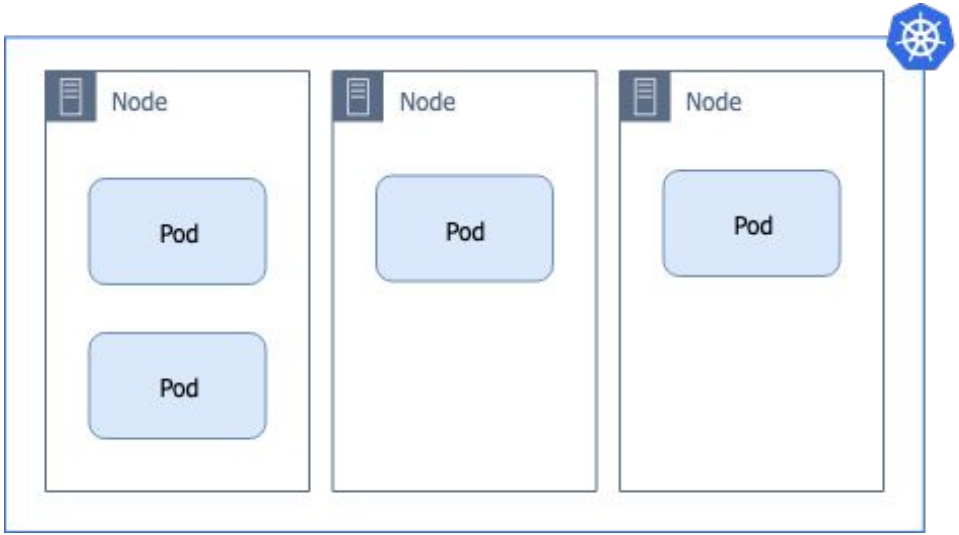
Manuel Zapf

Head of Product Open Source

@manuel_zapf

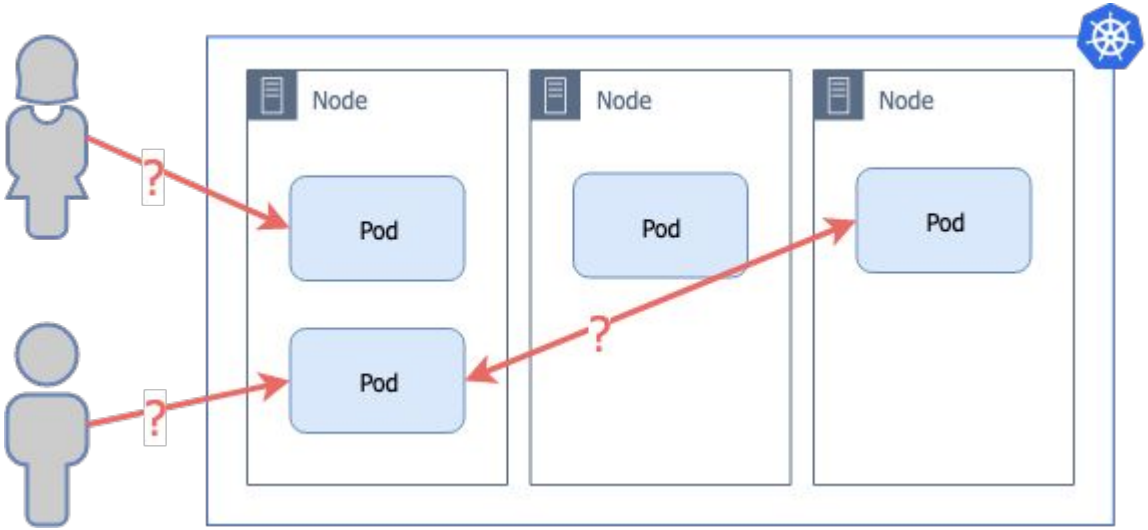
- Open Source Advocate @ Traefik Labs
- Software Engineering
  Go / NodeJS / Python / PHP
- Alles Container bezogenes
- Speaker
- Stolzer Papa (=

Ein klassisches Cluster



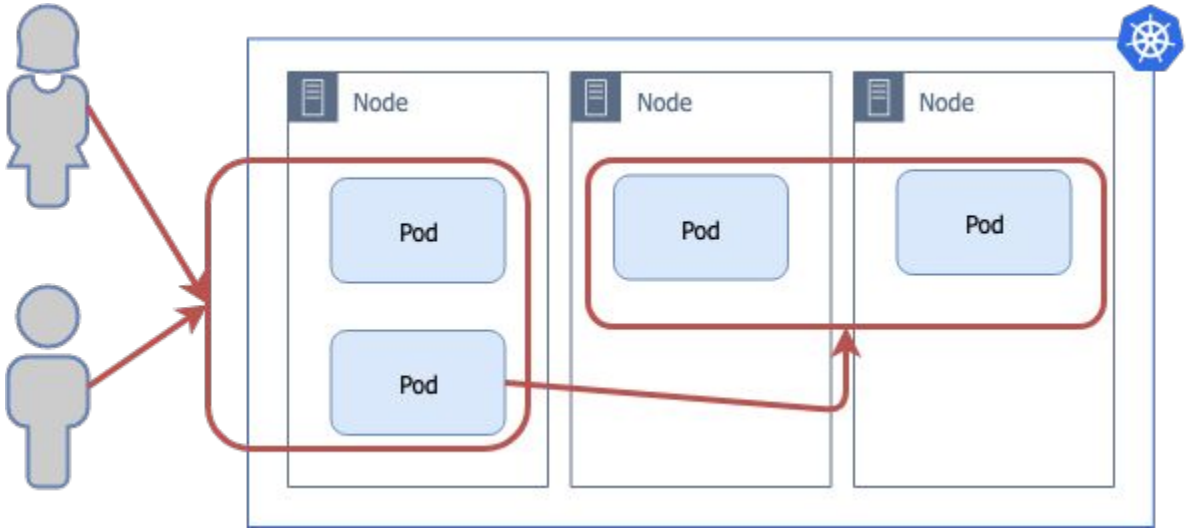
Ein klassisches Cluster

- Problem: Pods haben private Ips
- Wie kann ich Traffic zu Pods routen?
- Und wie zwischen Pods auf verschiedenen Nodes?



Services

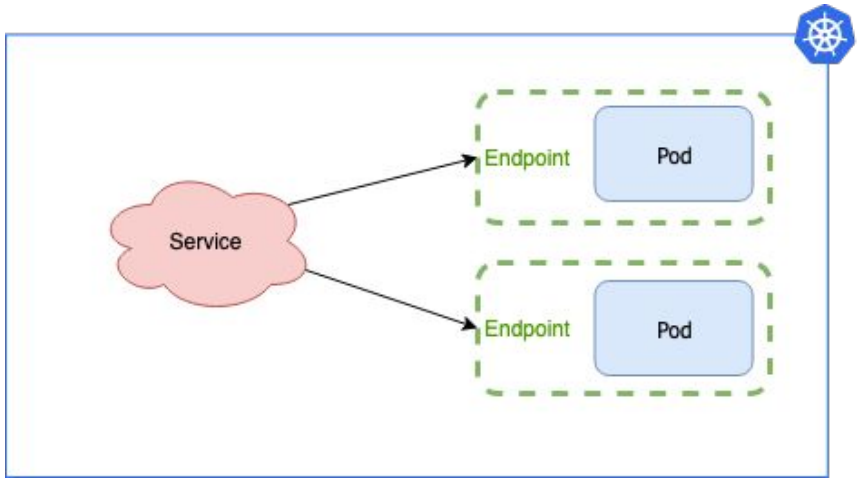
- Loesung: Services!
- Ziel: Pods im Cluster bekannt machen, so dass sie Traffic erhalten koennen



Services

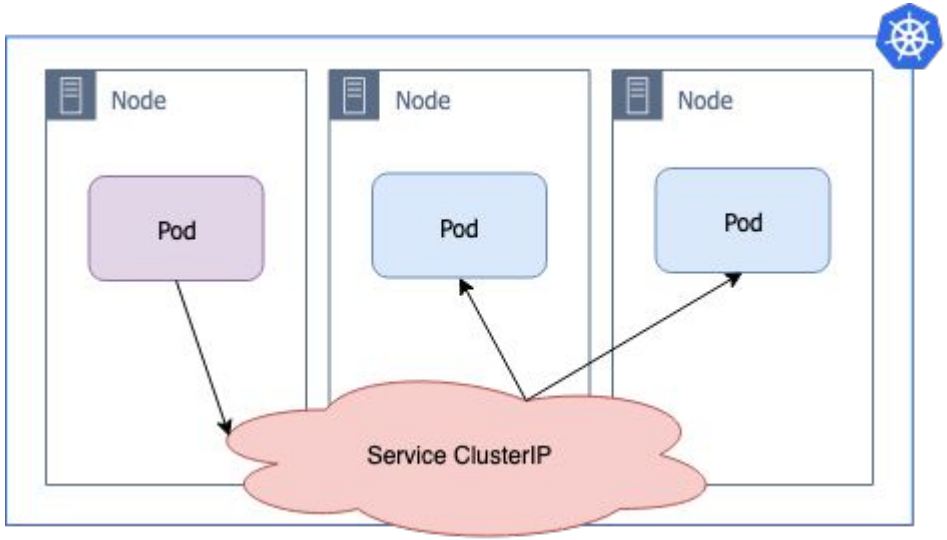
- Services koennen Loadbalancer sein!
- Services haben multiple Endpoints
  - Ausnahme: External name Services
- Endpoints werden durch die Kubernetes API entschieden

- Verschiedene Service Types
- Innerhalb eines Clusters
  - ClusterIP (default)
- Ausserhalb eines Clusters
  - NodePort
  - LoadBalancer



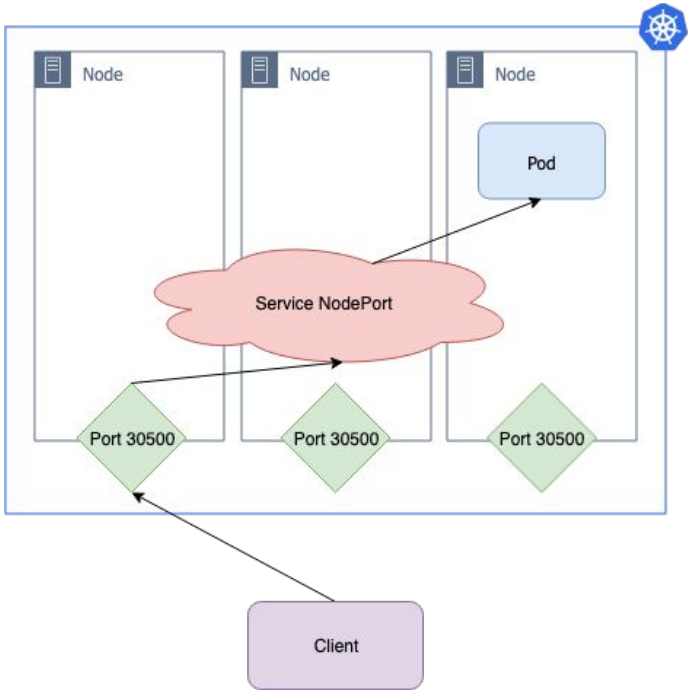
Services: Cluster IP

- Virtual IP
- Private im Cluster
- Z.b. Fuer Node zu Node



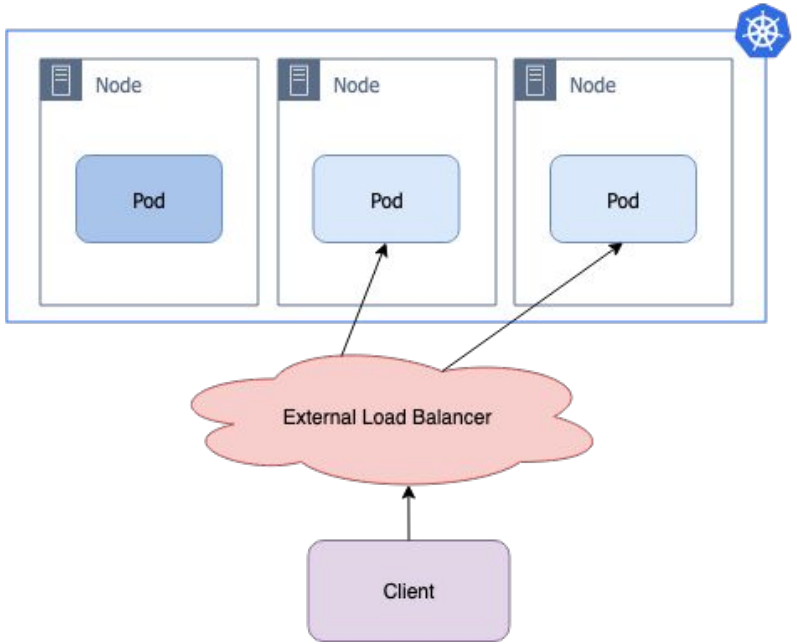
Services: NodePort

- Public Node IP und Port
- Wie ein “Routing Grid”



Services: LoadBalancer

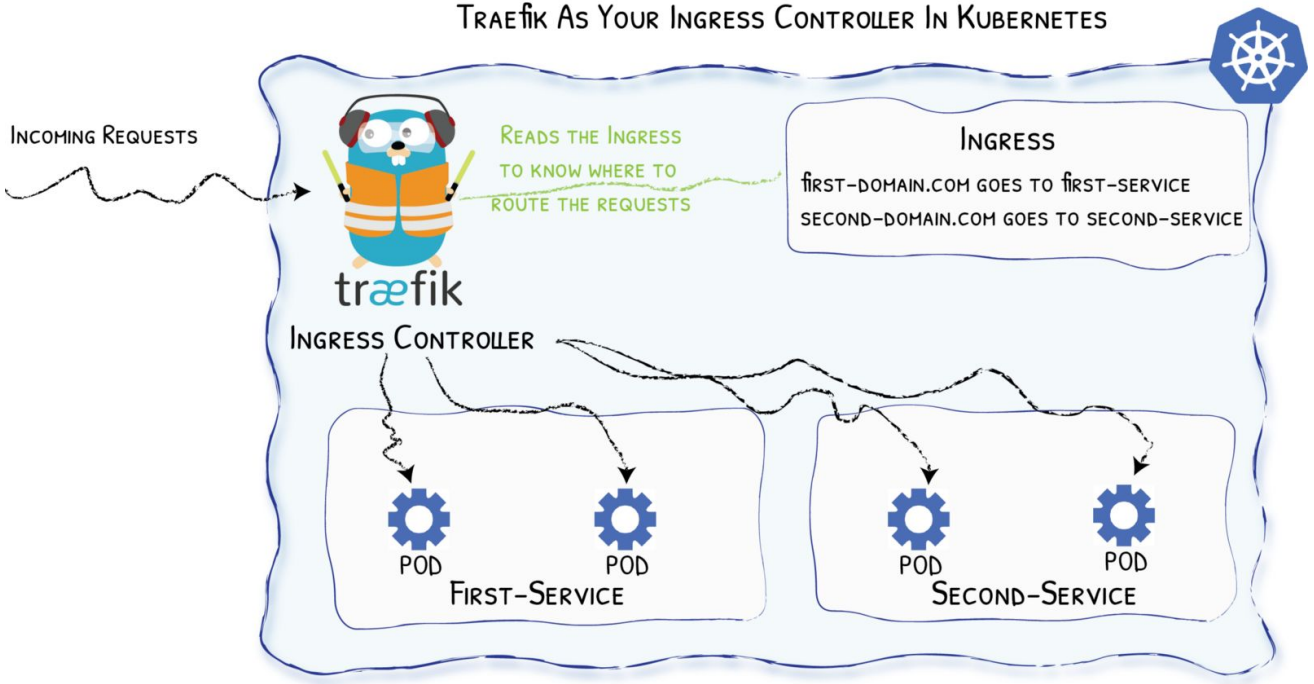
- Wie Nodeport
- Unterschied: Provisioniert einen externen LoadBalancer



Services

- Kontext: Mehrere Microservices extern erreichbar machen
- Herausforderung: Overhead durch mehrere LB fuer jede App
  - 1 LB Ressource pro App (Maschine oder Virtuell)
  - Mind. eine Public IP
  - DNS Nightmare
  - Keine Zentralisierung von Logs o.ae.
  - ClusterIP (default)

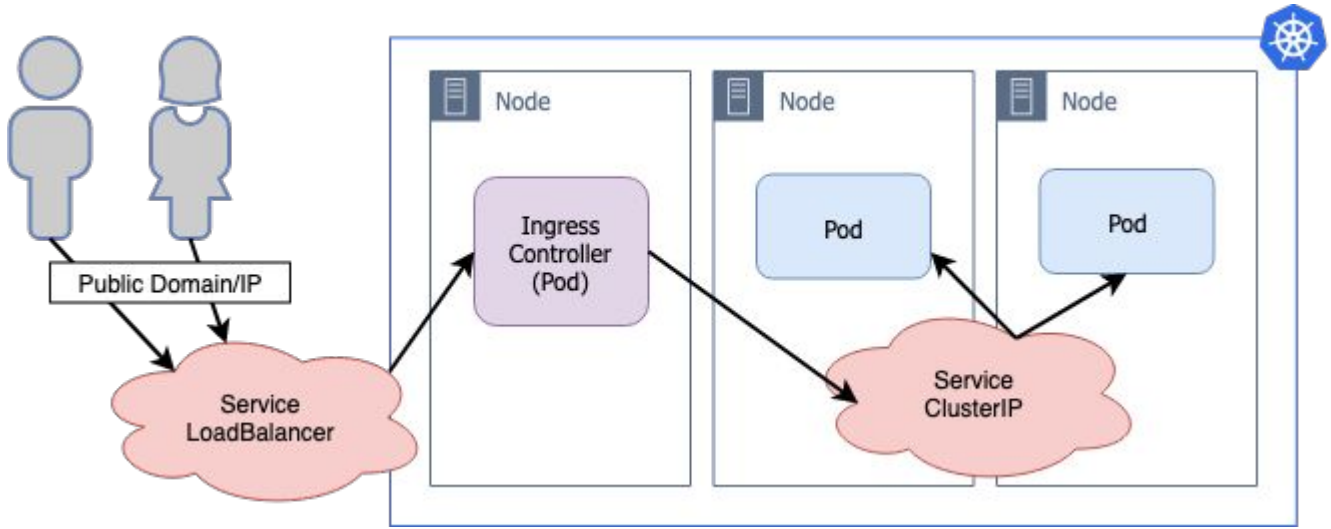
Dann kam der Ingress



Ingress

- Ingress Controller sind standard Kubernetes Applications
- Deployed als Deamonsets oder Pods
- Extern erreichbar gemacht durch Services
  - Bennoetigt immer noch eine Public IP und Co
  - Aber im besten Falle nur eine ;-)

Ingresses haben auch Services



Warum?

- Deutlich einfacheres Setup
- Nur ein Einstiegspunkt
- Weniger Konfiguration und messbarkeit
- Seperation of Concerns: Anderes LoadBalancing fuer externen Traffic etc

Limits?

- Designed fuer einfache Anwendungsfaelle
  - HTTP(S)
  - Virtual Host fokussiert
- Kein UDP / TCP
- Nur ein Ingress Controller?

tdlr

- Kubernetes gibt euch die Freiheit
  - Kein Problem fuer Multiple Ingress Controller
  - Kubernetes gibt euch Moeglichkeiten
- So viele Deployment Pattern...

Externe Load Balancer

- Ausserhalb von Kubernetes
  - Kommt allerdings auf die Plattform an (Infrastruktur / Cloud)
- Managed von K8s
  - Braucht Plugins fuer jeden LB Provider
  - Kein API oder kein K8s Support: Braucht NodePort
- Nenn mir deine K8s Distribution
  - Cloud
  - Semi-Cloud
  - Bare - Metal

Cloud Managed Kubernetes

- Cloud provider nutzen hauefig Ihre LB Loesugnen
  - Voll automatisiertes Management durch API
  - Grossartige User Experience
  - Profitiert von Cloud Provider HA und Performance
- Aber
  - Kosten
  - Configuration ist meist Cloud spezifisch (annotations)
  - Ist an die Limits des LB gebunden

Bare Metal Kubernetes

- Bester Ansatz: Metal LB, eine LB Implementierung in K8s
  - Nutzt alle K8s Ressourcen: HA, Deployment…
  - Layer 2 Routing und BGP
  - Aber… noch nicht offiziel Prod ready
- In anderen Faellen
  - Externer bzw. Statischer LB
  - Benoetigt aber einen Nodeport Service

Cloud Semi Managed Kubernetes

- Kommt auf die Compute power an: Cloud oder Bare Metal
- Man benoetigt ein weiteres Tool fuer die Cluster OPs
  - Kops
  - KubeAdm
    - Meisten dieser Tools managen aber schon einen LB

Cloud Semi Managed Kubernetes

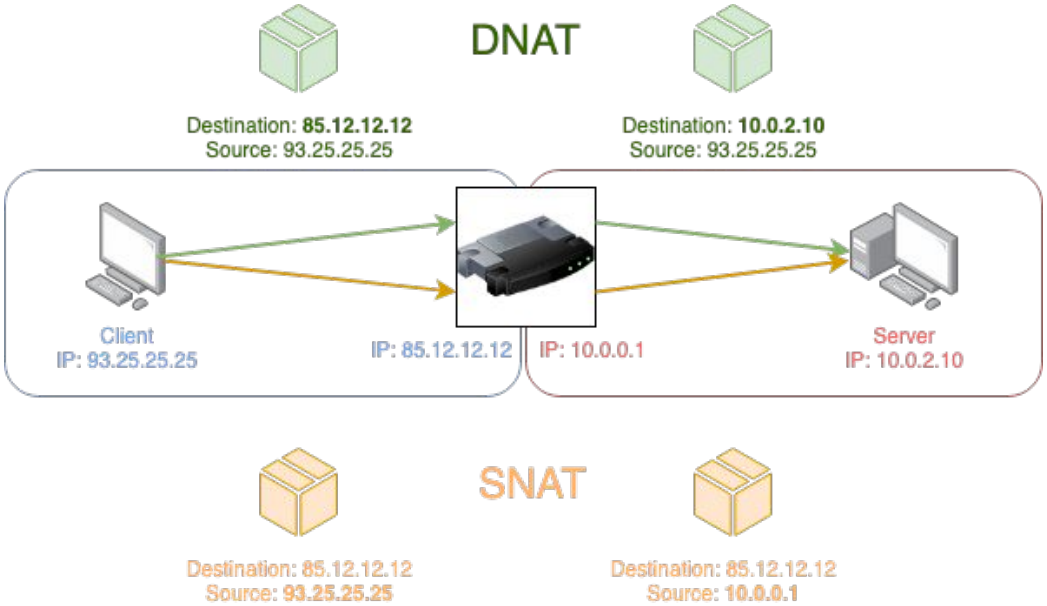
- Kommt auf die Compute power an: Cloud oder Bare Metal
- Man benoetigt ein weiteres Tool fuer die Cluster OPs
  - Kops
  - KubeAdm
    - Meisten dieser Tools managen aber schon einen LB

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 legals reasons and limit traffic in some cases.

NAT/DNAT/SNAT

- NAT = Network Address Translation
  - Ipv4 Welt: Router verstecken Ips, um Routing aus anderen Netzwerken zu erlauben
- DNAT = Destination NAT
  - Verschleiert die Ziel IP mit der internen Router IP
- SNAT = Source NAT
  - Verschleiert die Quel IP mit der internen Router IP

NAT/DNAT/SNAT



Wichtiges Ziel: Source IP erhalten!

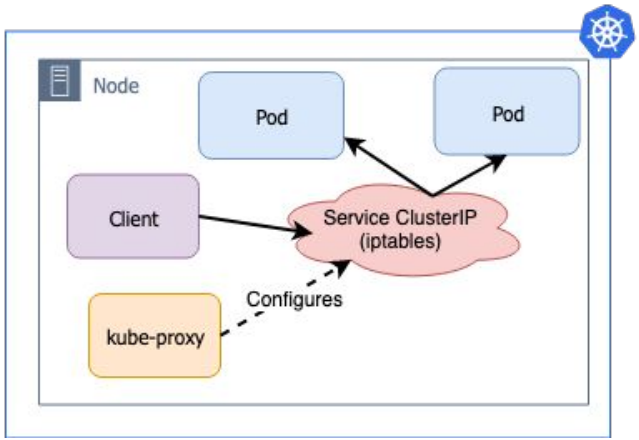
- Regel: SNAT verhindern
- Herausforderung: Aller verschiedenen Kompoennten koennten die Source IP verlieren und dazwischen funken

Innerhalb K8s: Kube Proxy

- Kube-proxy ist eine K8s Komponente welche auf jeder Worker Node laeuft
- Aufgabe: Virtuelle Service IPs managen
- Herausforderung Source IP: kube-proxy kann SNAT
- SNAT fuer kube-proxy kommt auf den Service an
    - Lets take a tour!

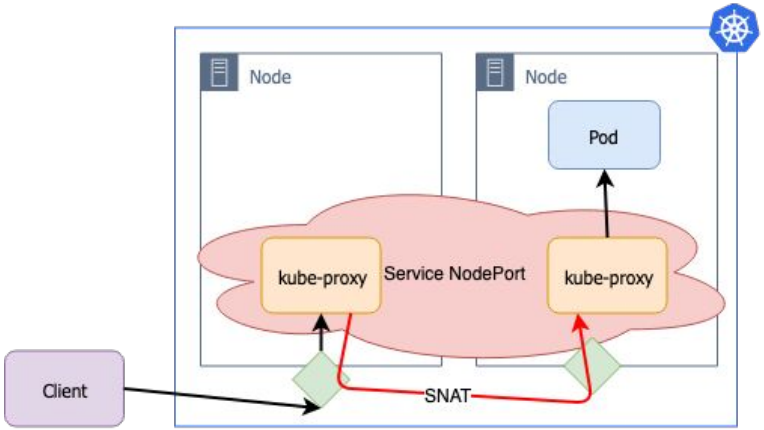
Source IP mit ClusterIP Service

- When kube-proxy is in “iptables” mode : kein snat ✔️
  - Das ist der default mode
- Keine weitere Komponente



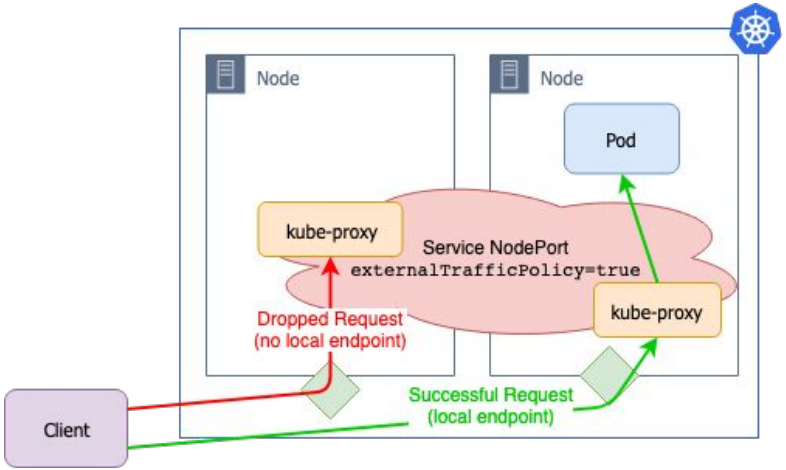
Source IP mit Service Node Port (default)

- SNAT findet statt ❌(wenn zur node des ziel pods geroutet wird)
  - Erstes Node zu Node Routing durch das Nodes network
  - Danach: Node zu pod durch das Pod netzwerk



Source IP mit Service Node Port (Local Endpoint)

- Kein SNAT ✔️ wenn `externalTrafficPolicy` = Local konfiguriert ist
- Downside: Gedroppte Request wenn auf der Ziel node kein entsprechender Pod laeuft



Source IP mit Service LoadBalancer (default)

- SNAT ❌ wie bei Nodeport
  - Externer LB kann auf jede Node Traffic routen
  - Wenn kein lokaler Endpoint verfügbar:
    - Node to Node mit SNAT

Source IP mit Service LoadBalancer (local Endpoint)

- kein SNAT ✔️ für LoadBalancer die local externalTrafficPolicy implementieren
  - GKE/GCE LB, Amazon NLB, etc.
  - 🛠️Nodes ohne local endpunkt werden vom LB entfernt aufgrund von fehlschlagenden Healthchecks
  - 👍🏾Pros: Keine gedroppten Requests von “aussen”
  - 👎🏾Cons: Verlässt sich auf die LB Healthchecks

SNAT Alternativen

- Manchmal geht es nicht ohne SNAT
  - Externer LB
  - Netzwerk constraints
  - Ingress Controller in der Mitte

- Netzwerk besteht aus mehreren Layern
  - Wenn HTTP: Source IP aus den Headern
  - Wenn TCP / UDP: Proxy Protocol
  - Oder: Distributed Logging / Tracing

HTTP Protocol Header

- X-Forwarded-From beeinhaltet eine koma seperarierte Liste von allen Source IP Snats die aufgetreten sind
  - ✔️ Wenn der externe LB oder Ingress Controller diesen Header beachten
  - ⚠️ Kein standard (header mit X- beginnend) was dazu führen kann, dass dieser nicht überall beachtet wird
    - Upcoming Official HTTP Header Forwarded

Proxy Protocol

- Vorgestellt von HAProxy
- Findet auf Layer 4 (Transport) statt fur TCP/UDP
- Ziel: "Proxies / Reversies Proxies hintereeinander schalten ohen Source zu verlieren"
- Von vielen Systemen unterstuetzt: AWS ELB, Traefik, Apache, Nginx, Varnish, etc.
- Anwendungsmuster: When SNAT stattfindet aber HTTP nicht moeglich ist

Distributed Logging & Tracing

🛠️ Idea:

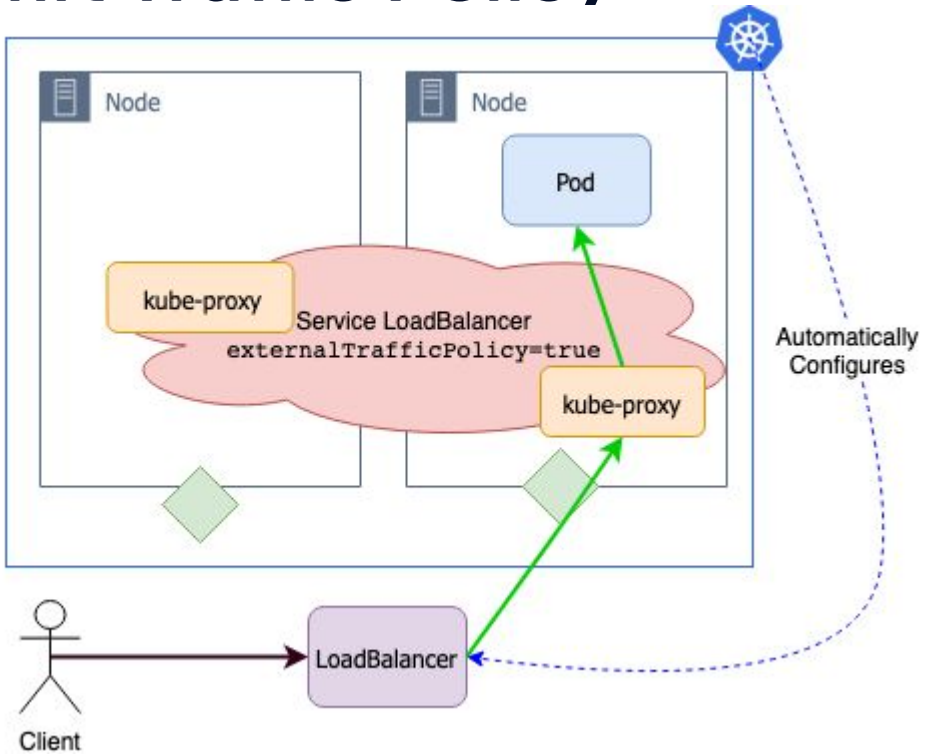
- Die Quell IP so schnell wie moeglich in einem Distributed Logging System ablegen
- Distributed Tracing nutzen um Request wieder zu finden

👍🏽Pros: kein complexes networking setup, distributed logging / tracing oft bereits im Cluster verfuegbar

👎🏽Cons: Starkes verlassen auf distributed logging / tracing

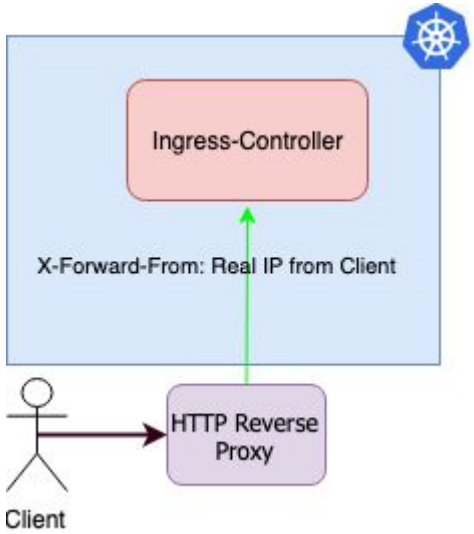
Externer LB mit Traffic Policy

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



HTTP Header

- 👍🏾Pros: Vereinfachtes Setup
- 👎Cons: Funktioniert nur mit 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

Fragen?

twitter @manuel_zapf
or https://community.traefik.io