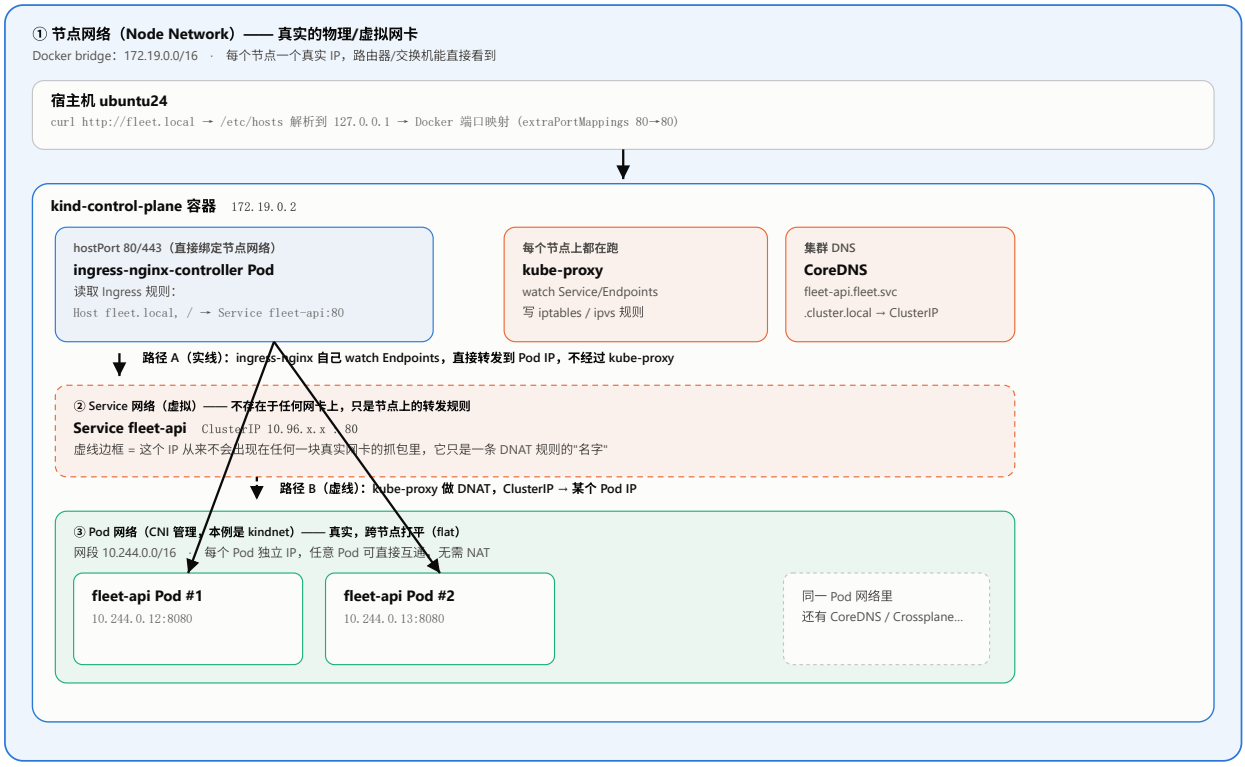
Kubernetes 内部网络架构

以你现在这套 kind 单节点集群为例：client → fleet.local → ingress-nginx → fleet-api Service → Pod

三层网络地址空间

K8s 里同时存在三种性质完全不同的"网络"，很多困惑都来自把它们当成一回事。



节点网络 —— 真实网卡，能抓包看到　　Service 网络 —— 纯虚拟，只是每个节点上的一条转发规则

Pod 网络 —— CNI 维护的叠加网络，跨节点打平、真实存在

实线箭头 = 路径①（Ingress 直连）　虚线箭头 = 路径②（Service DNAT）

两条不同的访问路径 —— 关键区别在这里

同样是"访问 fleet-api"，走 Ingress 和走集群内部 Service 调用，实际经过的组件不一样。

	路径 A：外部走 Ingress（你现在这次）	路径 B：集群内部 Pod 互相调用
发起方	curl fleet.local（宿主机）	另一个 Pod 里的代码
寻址方式	直接命中 hostPort → ingress-nginx	先查 CoreDNS 拿 ClusterIP
到 Pod IP 这一步	ingress-nginx 自己 watch Endpoints，直接转发到 Pod IP，不经过 kube-proxy	发往 ClusterIP，被 kube-proxy 的 iptables/ipvs 规则 DNAT 成 Pod IP
途经组件	ingress-nginx → Pod 网络	CoreDNS → Service(虚拟) → kube-proxy → Pod 网络

这也是为什么 Ingress 控制器需要单独的 RBAC 权限去 `watch` Service 和 Endpoints 对象——它是自己做负载均衡决策，而不是依赖 kube-proxy 帮它转发。

NetworkPolicy 加在哪一层

前面提到的 K8s 最佳实践里的 NetworkPolicy，管的是 ③ **Pod 网络** 这一层——本质是让 CNI 插件在 Pod 之间的流量路径上加一道防火墙规则（哪些 Pod 之间的直连允许，哪些拒绝），跟①②两层无关。也正因为这样，NetworkPolicy 的实际生效与否，完全取决于你的 CNI 插件是否支持它——kind 默认的 kindnet 就不支持，装了也不会生效，这也是为什么很多 kind 集群里 NetworkPolicy 顶多是个摆设。