Core Networking & Kubernetes Concepts for Software Engineers

HTTP (Hypertext Transfer Protocol)

The protocol used by web browsers and APIs to communicate. Example: When visiting https://example.com, your browser sends an HTTP request and gets a response. In Kubernetes, HTTP is used for most app and service communication.

IP Address

A unique numeric identifier for devices on a network. Kubernetes assigns each Pod and Service its own IP. Example: 10.244.0.3 is a typical Pod IP.

Hostname

A human-readable name mapped to an IP address. Example: google.com. Kubernetes uses DNS to resolve hostnames like my-service.default.svc.cluster.local.

DNS (Domain Name System)

Resolves hostnames to IP addresses. In Kubernetes, it maps Service names to internal IPs for communication.

Proxy

A server that forwards requests between clients and servers. Kubernetes uses proxies (like Ingress controllers) to manage and route traffic to services.

Pod

The smallest deployable unit in Kubernetes. A Pod wraps one or more containers, and gets a unique IP. Services connect to Pods, not users directly.

Cluster

A group of machines (nodes) running containerized apps in Kubernetes. Consists of control plane (master) and worker nodes where Pods run.

Core Networking & Kubernetes Concepts for Software Engineers

Load Balancer

Distributes incoming traffic across multiple servers or Pods. Kubernetes supports LoadBalancer-type services and Ingress for advanced HTTP routing.

Proxy Server

A Proxy Server sits between clients and backend services to forward requests, filter traffic, add security, or distribute load. It hides internal infrastructure, enforces policies, or provides caching. In Kubernetes, Ingress Controllers (like nginx) function as reverse proxies to route traffic to Services based on rules.

How It All Connects in Kubernetes

[Browser] -> http://my-online-store.com

|

| (DNS resolves hostname)

|

[Ingress Controller / Load Balancer]

|

[Service]

|

[Pod(s) running app]