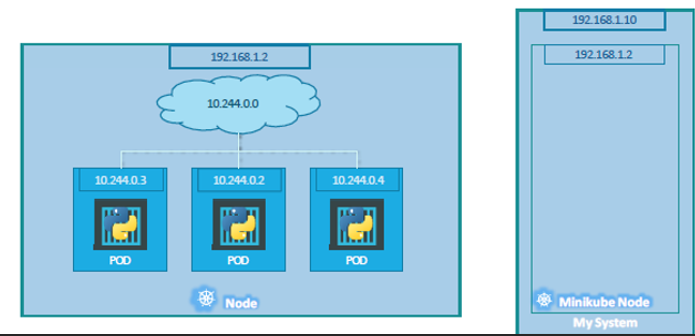
Kubernetes Networking



Kubernetes networking enables communication between containers, pods, services, and external systems.

**🔷 1. Basic Networking Concepts in Kubernetes**

We will start with a single node Kubernetes cluster.

* The node has an IP address, say it is 192.168.1.2 in this case. This is the IP address we use to access the Kubernetes node.
* All containers in a Pod share the same network namespace.
* Remember if we are using a Minikube setup, then we are talking about the IP address of the minikube. Our laptop may be having a different IP like 192.168.1.10.
* Unlike in the docker world where an IP address is always assigned to a Docker Container, in Kubernetes the IP address is assigned to a POD.
* Each POD in Kubernetes gets its own internal IP Address. In this case it’s in the range 10.244. series.

1. So how is it getting this IP address?
   * When Kubernetes is initially configured it creates an internal private network with the address 10.244.0.0 and all PODs are attached to it.
   * When we deploy multiple PODs, they all get a separate IP assigned.

The PODs can communicate to each other through this IP. But accessing other PODs using this internal IP address is not a good idea as its subject to change when PODs are recreated. We will see BETTER ways to establish communication between PODs.

A screenshot of a computer

AI-generated content may be incorrect.

* In this case we have two nodes running Kubernetes and they have IP addresses 192.168.1.2 and 192.168.1.3 assigned to them. Note that they are not part of the same cluster yet.
* Each of them has a single POD deployed. These pods are attached to an internal network, and they have their own IP addresses assigned.
* However, if we look at the network addresses, we can see that they are the same. The two networks have an address 10.244.0.0 and the PODs deployed have the same address too.
* This is NOT going to work well when the nodes are part of the same cluster. The PODs have the same IP addresses assigned to them and that will lead to IP conflicts in the network. Now that’s ONE problem.
* When a Kubernetes cluster is SETUP, Kubernetes does NOT automatically setup any kind of networking to handle these issues.
* As a matter of fact, Kubernetes expects us to setup networking to meet certain fundamental requirements.
* Some of these are that all the containers or PODs in a Kubernetes cluster MUST be able to communicate with one another without having to configure NAT. All nodes must be able to communicate with containers and all containers must be able to communicate with the nodes in the cluster. Kubernetes expects us to setup a networking solution that meets these criteria.

**🔷 2. Kubernetes Networking Components**

**✅ Pod-to-Pod Communication**

* Each pod gets a unique IP.
* Pods on different nodes can communicate **directly** via the network by the **CNI (Container Network Interface)** plugin.

**✅ Container-to-Container Communication (within a Pod)**

* All containers in a pod share the **same network namespace**.
* They communicate via localhost.

**✅ Pod-to-Service Communication**

* Services in Kubernetes define a stable IP and DNS name.
* Internally uses **iptables**, **IPVS**, or **eBPF** rules to forward traffic to a set of backend pods.
* kube-proxy handles this logic.

**✅ Service-to-External Communication**

* Pods can make outbound connections to the internet.
* Ingress and LoadBalancer objects allow **external traffic into the cluster**.

**🔷 3. Key Networking Resources**

**1. Service**

A stable endpoint to access a group of pods.

Types of Services:

| **Type** | **Description** |
| --- | --- |
| ClusterIP | Default. Only accessible within the cluster. |
| NodePort | Exposes service on static port on each node's IP. |
| LoadBalancer | Uses cloud provider to expose externally via a public IP. |
| ExternalName | Maps service to an external DNS name. |

**2. Ingress**

* Manages external HTTP/HTTPS traffic to services.
* Uses **Ingress Controllers** (like NGINX, Traefik).
* Provides routing, TLS, and virtual hosting.

**3. NetworkPolicy**

* Defines rules for how Pods can communicate.
* Controls **traffic flow** at the IP address or port level.

**🔷 4. How Communication Happens**

**🧩 Example: Pod A talks to Pod B**

1. Pod A (on Node 1) wants to talk to Pod B (on Node 2).
2. Pod A uses Pod B’s IP (or via a service).
3. Network traffic is routed through the CNI plugin.
4. Network plugins like Calico, Flannel, or Cilium handle cross-node routing.

A logo of a cloud

AI-generated content may be incorrect.

**🔷 5. CNI (Container Network Interface)**

* Kubernetes doesn’t implement its own networking.
* And we don’t have to set it up all on our own as there are multiple pre-built solutions available.

| **Plugin** | **Features** |
| --- | --- |
| Flannel | Simple overlay network. Easy to configure. |
| Calico | Supports network policies, BGP, and NAT-less routing. |
| Cilium | Uses eBPF for high performance and observability. |
| Weave | Mesh networking. Simpler for small clusters. |

**🔷 6. DNS in Kubernetes**

* Kubernetes has an internal DNS server (CoreDNS) that:
  + Provides name resolution for services and pods.
  + Each service gets a DNS name: myservice.mynamespace.svc.cluster.local.

**🔷 7. kube-proxy**

* A daemon that runs on each node.
* Maintains rules (via iptables, IPVS, or eBPF) to route traffic to the appropriate pod.
* Works with services to ensure load balancing and availability.

A screenshot of a computer

AI-generated content may be incorrect.

* With the Calico networking setup (or any), it now manages the networks and IPs in my nodes and assigns a different network address for each network in the nodes.

**🔷 Summary**

| **Concept** | **Description** |
| --- | --- |
| Pod networking | Each pod gets an IP, can talk to others without NAT |
| CNI plugins | Provide IP management and routing |
| Services | Abstract access to pods |
| Ingress | HTTP(s) routing to services |
| NetworkPolicy | Security and traffic rules |
| kube-proxy | Routes service traffic to pods |
| CoreDNS | DNS service for internal name resolution |