


Networking






What is Network?

A network is a group of two or more computers or other electronic devices that are interconnected for the purpose of exchanging data and sharing resources.

What is a networking model?



The networking model describes the architecture, components, and design used to establish communication between the source and destination systems. Aliases for network models include protocol stacks, protocol suites, network stacks, and network protocols.



There are 2 predominant models available.

1. Open Systems Interconnection (OSI) Model
2. Transmission Control Protocol/Internet Protocol (TCP/IP) Model

Network Models



TCP/IP MODEL	OSI MODEL	PROTOCOLS
Application Layer	Application Layer	FTP,HTTP,Telnet
	Presentation Layer	JPEG,MPEG
	Session Layer	NFS,SQL,PAP
Transport Layer	Transport Layer	TCP,UDP
Network Layer	Network Layer	IPv4,IPv6
Network Access Layer	Data Link Layer	ARP,CDP,STP
	Physical Layer	Ethernet,Wi-Fi

Kubernetes Networking

Networking is a central part of Kubernetes, but it can be challenging to understand exactly how it is expected to work.

Since a Kubernetes cluster consists of various nodes and pods, understanding how they communicate between them is essential. The Kubernetes networking model supports different types of open-source implementations.

The **Kubernetes networking model**, on the other hand, natively supports multi-host networking in which pods are able to communicate with each other by default.

The Kubernetes network model specifies:

- Every pod gets its own IP address
- Containers within a pod share the pod IP address and can communicate freely with each other
- Pods can communicate with all other pods in the cluster using pod IP addresses (without [NAT](#))
- Isolation (restricting what each pod can communicate with) is defined using network policies

Kubernetes network implementation

Kubernetes **does not provide a default network implementation**; it only enforces a model for third-party network implementations that needs to be plugged into Kubernetes using the **CNI (Container Network Interface) API**. There are lots of different kinds of CNI plugins

CNI (Container Network Interface)?

- Container Network Interface (CNI) is a framework for dynamically configuring networking resources.
- It uses a group of libraries and specifications written in Go.
- CNI has a multitude of supported plugins, with major container orchestration frameworks like Kubernetes having implemented it. **Plugins** address various container networking functions and **must conform to CNI standards defined by the CNI specification**.
- CNI offers specifications for multiple plugins **because networking is complex, and user needs may differ**. It is essential to choose the right plugins for your project and use case.

CNI Plugins

Flannel - a very simple overlay network that satisfies the Kubernetes requirements. Flannel runs an agent on each host and allocates a subnet lease to each of them out of a larger, preconfigured address space. Flannel creates a flat network called as overlay network which runs above the host network.

Project Calico - an open-source container networking provider and network policy engine. Calico provides a highly scalable networking and network policy solution for connecting Kubernetes pods based on the same IP networking principles as the internet. Calico can be deployed without encapsulation or overlays to provide high-performance, high-scale data center networking.

Weave Net - a cloud native networking toolkit which provides a resilient and simple to use (does not require any configuration) network for Kubernetes and its hosted applications. It provides various functionalities like scaling, service discovery, performance without complexity and secure networking.

Other options include **Cisco ACI** , **Cilium** , **Contiv** , **Contrail** , **Kube-router** , **Nuage** , **OVN** , **Romana** , **VMWare NSX-T with NSX-T Container Plug-in (NCP)** .