INTRODUCTION TO KUBERNETES NETWORKING

In this blog post, we will introduce Kubernetes networking. This is meant to be an introduction into Kubernetes networking and I assume you have a basic understanding of Kubernetes pods and Kubernetes service and their use cases.

Communication in K8s

Before we can understand Kubernetes networking, we must first understand what we are trying to solve. There are 3 issues we need to solve in a Kubernetes cluster:

1. Container to Container communication
2. pod to pod communication
3. pod to service communication

Solving for Kubernetes Network Issues

Let us go over each one of the bullet points we mentioned above and how Kubernetes attempts to solves it.

Container to Container Communication

This is solved with the concept of *Pods*. In Linux, a network namespace contains its own network resources, i.e interfaces, routing tables, etc. This same concept applies to Pods. When a pod is created, it gets a network namespace that the containers in the pod will share resources therefore allowing all containers in a pod to communicate through localhost.

Pod to Service Communication

A service is an abstraction that routes traffic to a set of pods. A service gets its own Cluster IP, the pods also get their own IP's and when requests need to reach the pods, it is sent to the service Cluster IP and then forwarded to the actual IP of the pods. This way pods(mortals) can come and go without worrying about updating IP's since by default the service will automatically update its endpoint with the IP of the pod that it targets as those pod changes IP. The way the service knows what pod to target is through LabelSelector. This is an important concept to be aware of since LabelSelector is the only way the service knows what pod to target and keep track of their IP's.

Pod to Pod Communication

For communication between pods, Kubernetes imposes the following fundamental requirements on any networking implementation:

- all Pods can communicate with all other Pods without using *network address translation (NAT)*
- all Nodes can communicate with all Pods without NAT.
- the IP that a Pod sees itself as is the same IP that others see it as.

K8s networking tools

There are plenty of tools we can add to Kubernetes to help us meet these requirements:

- Cilium is open source software for providing and transparently securing network connectivity between application containers.
- Flannel is a very simple overlay network that simply satisfies the Kubernetes requirements.
- Kube-router is a purpose-built networking solution for Kubernetes that aims to provide high performance and operational simplicity.
- Plenty other solid projects to use!

The idea is that when a pod is created, Linux Virtual Ethernet Device or *veth pair* can be used to connect Pod network namespaces to the root network namespace.

- **Pods on the same node** can talk through a bridge. A Linux Ethernet bridge is a virtual Layer 2 networking device used to unite two or more network segments, working transparently to connect the two networks together.
- For **pods on separate nodes**, assuming connectivity between the nodes, the request will go from root namespace of one node to the root namespace of other node. The node should then know how to route traffic to the correct pod.

Additional resources

For more on Kubernetes, explore these resources:

- Kubernetes Guide, with 20+ articles and tutorials
- BMC DevOps Blog
- Bring Kubernetes to the Serverless Party
- How eBay is Reinventing Their IT with Kubernetes & Replatforming Program