Day4

Docker Network Model

```
- Each container gets an unique private IP on the system it runs
- Containers can be connected to Docker Network
- Docker creates a default bridge network called docker0 with subnet
172.17.0.0/16 ( 65535 IP addresses )
- Containers get their IP address from the subnet range assigned to the
Docker network
- Containers running on same network can communicate with each other
directly
- a single container can be connected to multiple networks, which means
they may get multiple IP addresses
```

Subnet

```
- 172.17.0.0/16
- IPV4 IP address
- 172 - 1 byte
- 17 - 1 byte
- 0 - 1 byte
- 0 - 1 byte
- IPv4 - 4 bytes ( 32 bits )
- CIDR - Classless Inter Domain Routing
- First IP in 172.17.0.0/16 - 172.17.0.0
- 172.17.0.1
- 172.17.0.255
- 172.17.1.0
- 172.17.1.255
- 256 x 256 = 65535 IP addresses are there in 172.17.0.0/16 Network
```

Kubernetes Network Model

```
- Kubernetes doesn't implement Network, it only publishes its Network
requirements as Kubernetes Network specification, while third-party vendors
implements the Kubernetes Network specification, which is referred as
Kubernetes Network Model
- Each Pod should get an unique cluster wide IP address within the
Kubernetes cluster
- Pod Network
  - all pods should be able to communicate with each other whether they run
in same node or different nodes
  - kubelet should be able to communicate with the pods that runs on the
node where kubelet is running
```

```
- Service Network
  - Pods are temporary as they get removed in the process of scale up/down, rolling update, etc.
  - application developers should not use Pod IP to access them
  - every service should get an unique cluster wide name and IP address
  - service IP once assigned to a service will not change until the service exists
  - i.e service IP is stable
  - service represents a group of Load Balanced Pods behind them
- Ingress
  - makes services accessible outside the Kubernetes cluster
- Network Policy
  - helps controlling communication between Pods
  - helps controlling Pod incoming/outgoing traffic
```

Kubernetes Network CNI Addons

```
- the Kubernetes Network Model(Specification) is implemented by third-party CNI plugins/addons
- Some of the Kubernetes CNI Network addons are
  - Calico
  - Weave Net
  - Flannel
  - Antrea
  - Canal
  - Cilium
  - Gateway API
  - Multus
  - Romana
  - Spiderpool
  - Nuage
  - NST-T
  - Knitter
  - OVN-Kubernetes
  - Nodus
  - Contrail
  - Contiv
  - ACI
- Popular addons most commonly used by many companies
  - Flannel
  - Calico
  - WeaveNet
```

Flannel

You may find this article interesting

```
https://mvallim.github.io/kubernetes-under-the-hood/documentation/kube-flannel.html
```

```
```

Let's understand flannel briefly

```
- one of the oldest and most mature CNI plugins available
- is a simple, lightweight layer 3 fabric for Kubernetes
- uses Overlay Network
- developed by CoreOS
- operates on Layer 3 of the OSI model and uses the VX-LAN as its default
backend to move network packets between nodes
- provides access to basic networking features and requires limited amount
of administration to set up and maintain
- supports a variety of backends like VX-LAN, host-gateway, AWS VPC,
AliVPC, IPIP, and IPSec etc.,
- overlay network is a network that is layered on top of another network
- overlay network can be used to handle pod-to-pod traffic between nodes
- Overlay networks work by encapsulating network packets
- when a pod initiates a connection to an IP address outside of the
cluster, the node hosting the pod will use SNAT (Source Network Address
Translation) to map the source address of the packet from the pod IP to the
node IP
- is a great entry level choice for Kubernetes cluster networking
- drawbacks
  - doesn't support Network Policy
  - as each packets are encapsulated by sender and de-encapsulated by the
receiver it impacts overvall network performance negatively
- when flannel CNI is installed in Kubernetes, one Flannel Pod gets created
on each Kubernetes node
- flannel either uses Kubernetes API to store network configurations in
etcd or gets direct write access to etcd database just like API Server
- when kubernetes master node is bootstrapped(installed), we need to assign
a Pod network subnet
- eg: kubeadm init --pod-network-cidr=10.244.0.0/16
- in the above example 10.244.0.0/16 ( 65535 IP addresses are allocated for
Pods created in the K8s cluster )
- the above subnet is sub-divided into small subnets(IP ranges) and
allocated to every node by Flannel
  - For example
    - Master 1 - 10.244.1.0/24 ( Upto 256 IP addresses, as Nodes by default
can support only 110 Pods this is more than enough )
    - Master 2 - 10.244.2.0/24
    - Master 3 - 10.244.3.0/24
    - Worker 1 - 10.244.4.0/24
    - Worker 2 - 10.244.5.0/24
    - Worker 3 - 10.244.6.0/24
- the subnet for every node is
```

Calico Overview

```
- Calico
  - implemented by company called Tigera
  - comes in 2 flavours
    - opensource and
    - enterprise
  - most popular and commonly used in Kubernetes/Openshift CNI
  - provides both Network and Network Policy
  - operates on Layer 3 of the OSI model and uses the BGP(Border Gateway
Protocol) protocol to move network packets between nodes
  - BGP is one of the fundamental building blocks of the internet, with
exceptional scaling characteristics
  - Using BGP, Calico directs packets natively, without needing to wrap
them in additional layers of encapsulation
```

Weave Net Overview

```
- is a flexible networking solution for Kubernetes/Openshift clusters
- developed by a company called WeaveWorks
- Weave comes in 2 flavours
  - opensource and paid
- weave routes packets using fast datapath method
- weave routes packets uses a slower network method called sleeve packet
forward when fast datapath fails
- is easy to install and configure
- creates a mesh overlay network to connect all the nodes in the cluster
- Weave is a good choice for organizations that need a flexible and
scalable networking solution for their Kubernetes/Openshift clusters
```