**Cluster Networking:**

Networking is a central part of Kubernetes, but it can be challenging to understand exactly how it is expected to work. There are 4 distinct networking problems to address:

1. Highly-coupled container-to-container communications: this is solved by Pods and localhost communications.

2. Pod-to-Pod communications.

3. Pod-to-Service communications.

4. External-to-Service communications

**Container to container communication inside pod**

Example:

kind: Pod

apiVersion: v1

metadata:

name: testpod

spec:

containers:

- name: c00

image: ubuntu

command: ["/bin/bash", "-c", "while true; do echo Hello-sagar; sleep 5 ; done"]

- name: c01

image: httpd

ports:

- containerPort: 80

**Commands:**

a. Kubectl apply -f filename.yaml

b. To continuously stream logs from pod testpod and container c00 -- > kubectl logs -f testpod -c c00

c. To execute an interactive bash or shell in the particular pod and container --> kubectl exec -it testpod -c c00 /bin/bash

d. Apt update && apt install curl

e. To attempt to access service running on port 80 --> Curl localhost:80

f. Now it will show the output of application which is running inside 2nd container.

**Pod to Pod Communication within same node Example**

kind: Pod

apiVersion: v1

metadata:

name: testpod1

spec:

containers:

- name: c00

image: ubuntu

command: ["/bin/bash", "-c", "while true; do echo Hello-sagar; sleep 5 ; done"]

---

kind: Pod

apiVersion: v1

metadata:

name: testpod2

spec:

containers:

- name: c01

image: httpd

ports:

- containerPort: 80

1. Pod to communication will happen via the Ips.
2. By default pod Ip will not accessible outside the node.

**Commands:**

1. Kubectl apply -f filename.yaml
2. Kubectl get all
3. kubectl exec -it pod/testpod1 -- /bin/bash
4. apt install && apt install curl
5. Ping IPaddressOFPod2:80