# **Pod To Pod Communication on Same Node Across the Node**

Practicing pod-to-pod communication in a Kubernetes environment involves setting up a cluster and deploying pods that can communicate with each other. Here’s a step-by-step guide to help you get started:

Prerequisites

Kubernetes Cluster: Ensure you have a Kubernetes cluster set up. You can use Minikube for local testing or a cloud provider like Google Kubernetes Engine (GKE), Amazon EKS, or Azure AKS.

kubectl: Make sure you have kubectl installed and configured to interact with your cluster.

Step 1: Setting Up the Kubernetes Cluster

If you don't have a Kubernetes cluster, you can create one using Minikube:

sh

Copy code

minikube start

Step 2: Creating a Namespace (Optional)

It’s a good practice to create a namespace for your resources to keep them organized.

sh

Copy code

kubectl create namespace pod-communication

Step 3: Deploying Pods

We'll deploy two simple pods that can communicate with each other. We will use Nginx as the application for one pod and BusyBox for the other.

Nginx Deployment:

Create a file nginx-deployment.yaml:

yaml

Copy code

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

namespace: pod-communication

spec:

replicas: 1

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

Apply the deployment:

sh

Copy code

kubectl apply -f nginx-deployment.yaml

BusyBox Deployment:

Create a file busybox-pod.yaml:

yaml

Copy code

apiVersion: v1

kind: Pod

metadata:

name: busybox

namespace: pod-communication

spec:

containers:

- name: busybox

image: busybox:latest

command: ['sh', '-c', 'sleep 3600']

Apply the pod:

sh

Copy code

kubectl apply -f busybox-pod.yaml

Step 4: Service Creation

Create a service for the Nginx deployment to expose it within the cluster.

Create a file nginx-service.yaml:

yaml

Copy code

apiVersion: v1

kind: Service

metadata:

name: nginx-service

namespace: pod-communication

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

targetPort: 80

Apply the service:

sh

Copy code

kubectl apply -f nginx-service.yaml

Step 5: Testing Pod-to-Pod Communication

Now, let’s test the communication from the BusyBox pod to the Nginx service.

Get a shell into the BusyBox pod:

sh

Copy code

kubectl exec -it busybox -n pod-communication -- sh

Inside the BusyBox pod, use wget or curl to access the Nginx service:

sh

Copy code

wget -qO- http://nginx-service

You should see the Nginx welcome page HTML content.