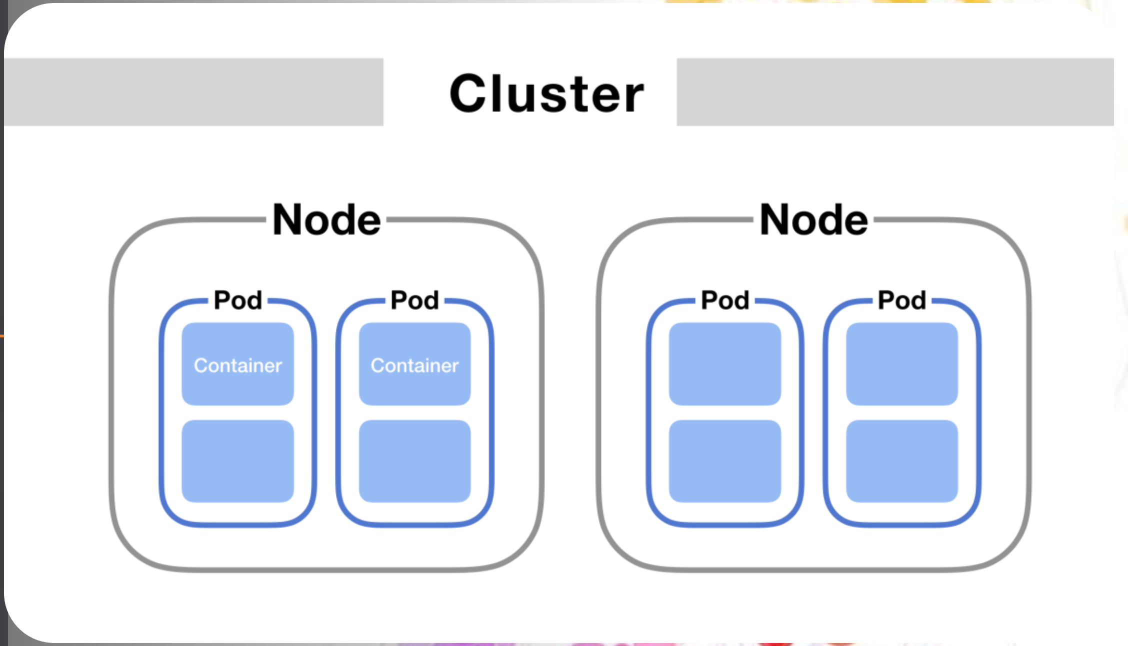
# Kubernetes Network Policies

Managing Network Access in Kubernetes

# Introduction

Network policies are crucial for controlling communication between pods in a Kubernetes cluster. They allow you to define which pods can communicate with each other and other network endpoints.

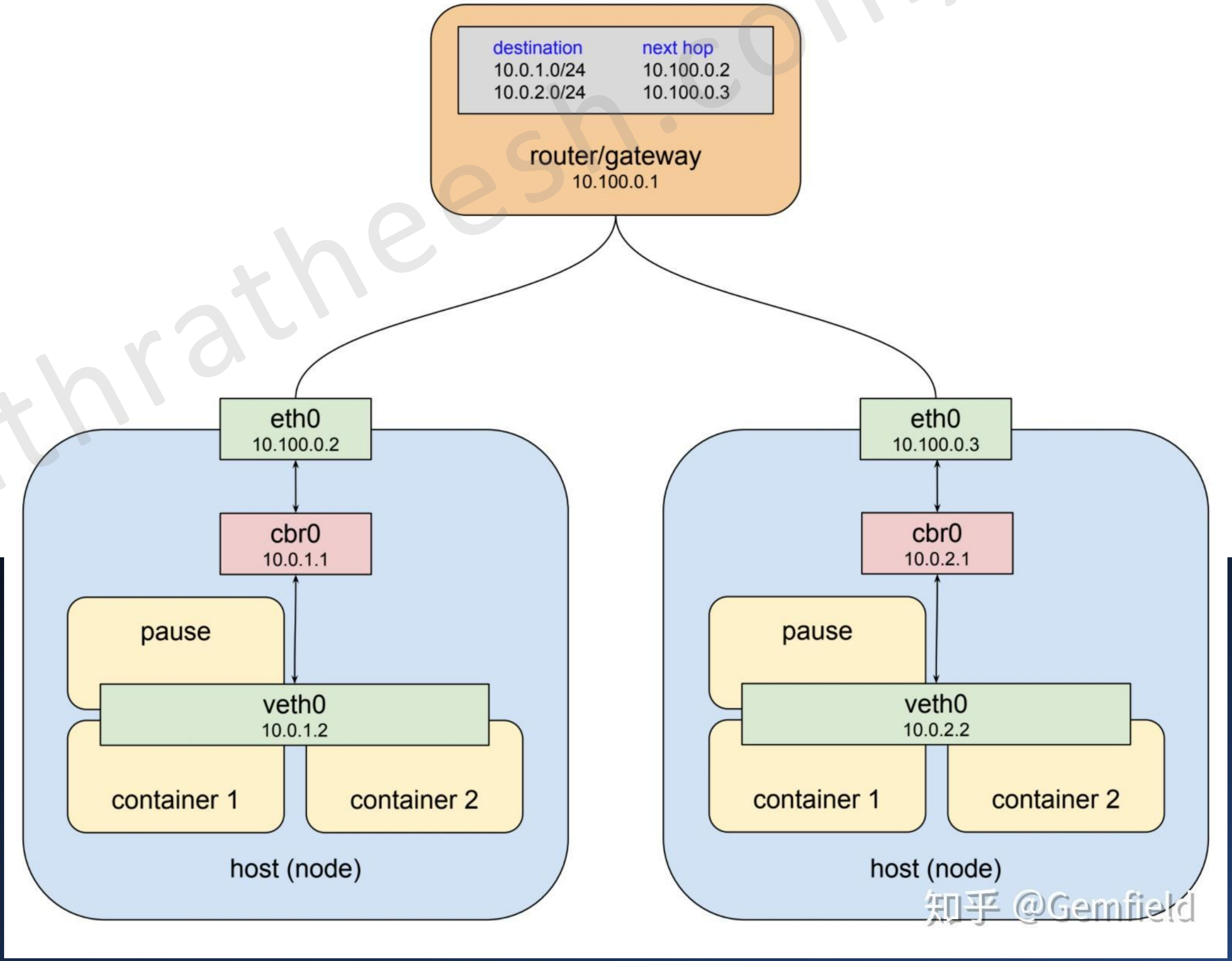


## Importance

In a microservices architecture, it's essential to have granular control over network traffic to ensure security, prevent unwanted access, and manage service dependencies.

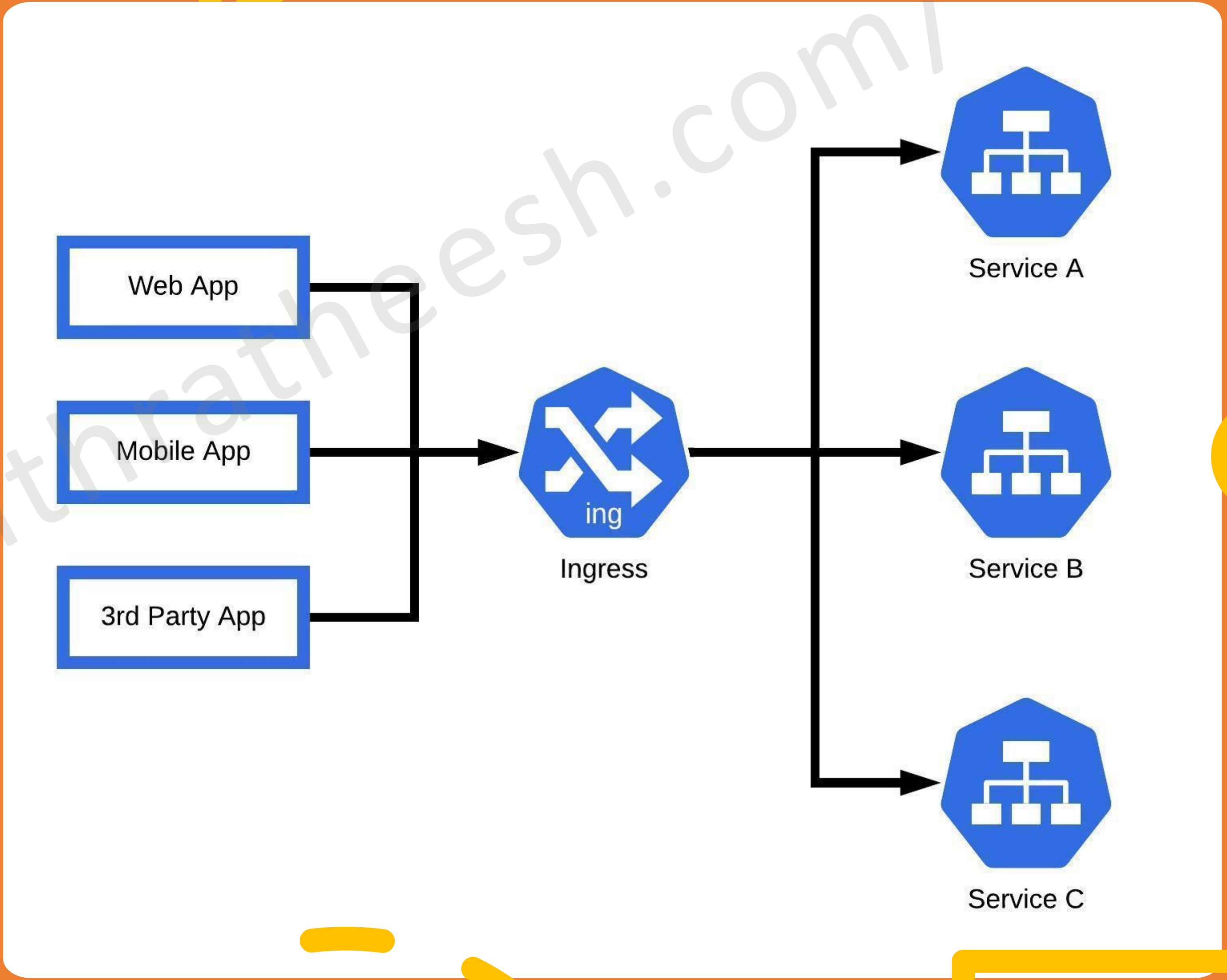
## Default Behaviour

By default, pods in a Kubernetes cluster can communicate with all other pods. A pod becomes isolated by having a Network Policy that selects it.



## Policy Types

Ingress controls incoming traffic to the pod, whereas Egress controls outgoing traffic from the pod. You can define policies for both or either.

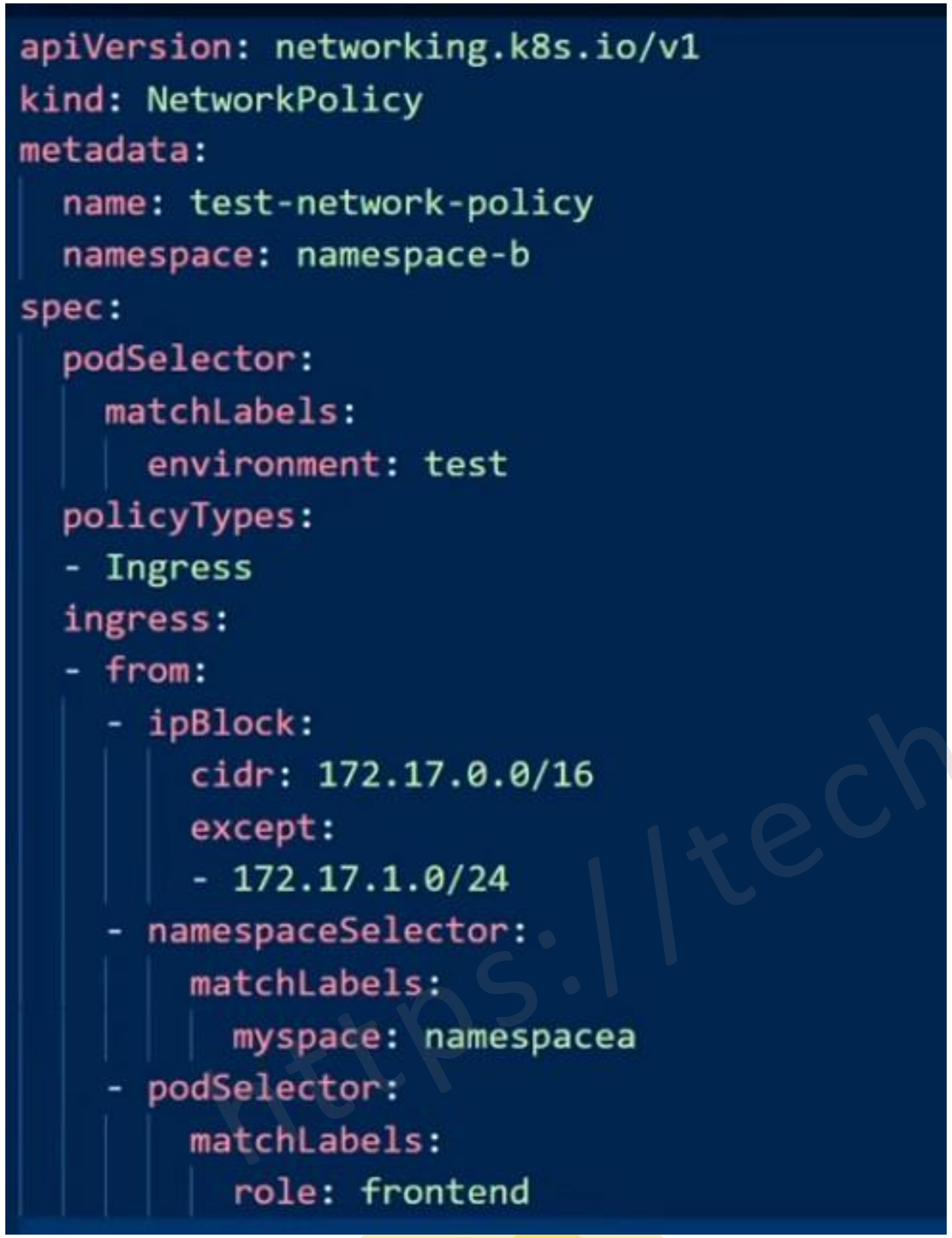


## Policy Components

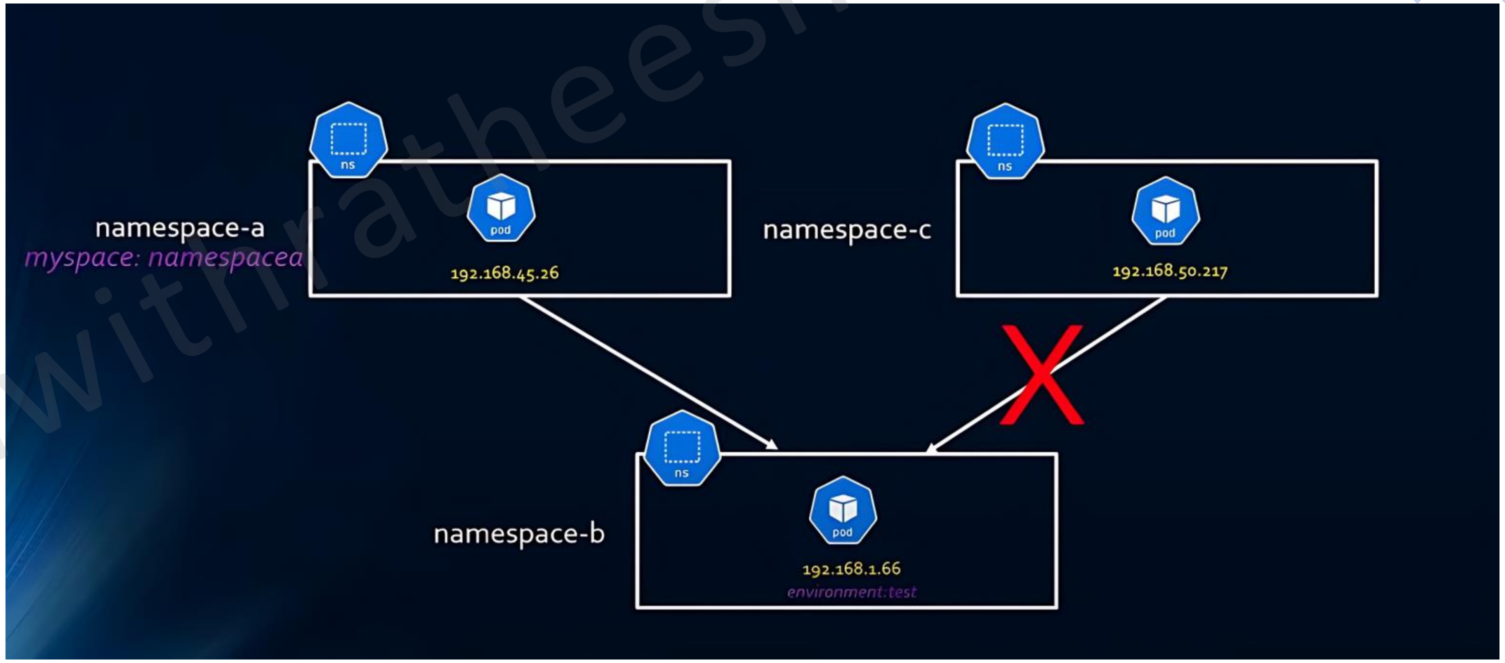
Network policies utilize pod selectors to target specific pods, define policy types, and set specific rules based on IP, ports, and more.

## Sample Ingress Policy

Display a basic YAML example of an Ingress policy and explain the specific components and their purpose.

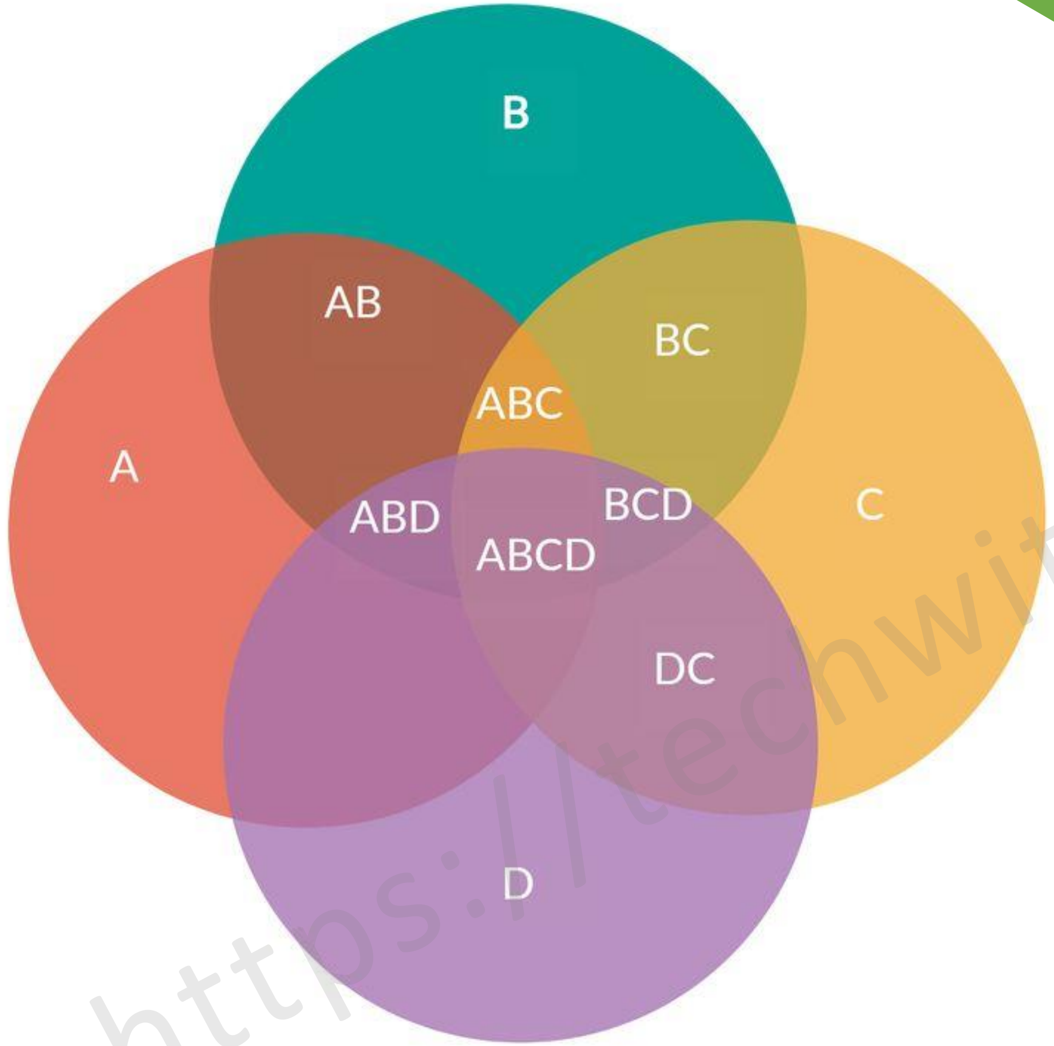


```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: namespace-b
spec:
  podSelector:
    matchLabels:
      environment: test
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          myspace: namespacea
    - podSelector:
        matchLabels:
          role: frontend
```



## Sample Egress Policy

Show a basic YAML example of an Egress policy. Highlight and explain its specific parts.

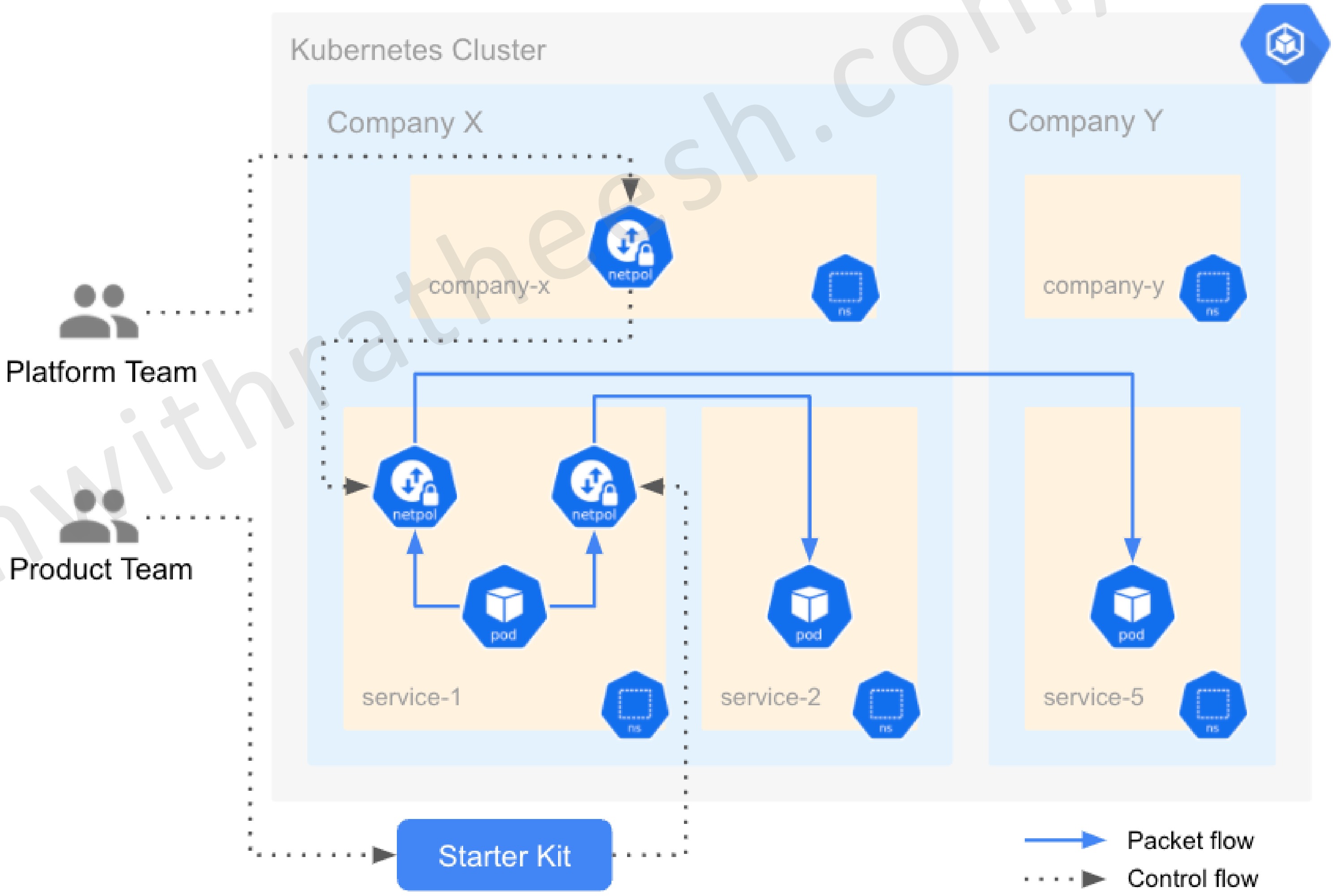


## Combining Policies

Multiple network policies can select the same set of pods. Policies are additive, and if any policy allows the traffic, it can flow.

# Namespace Specific Policies

Discuss how policies can be applied at the namespace level, allowing for broader controls over groups of pods.

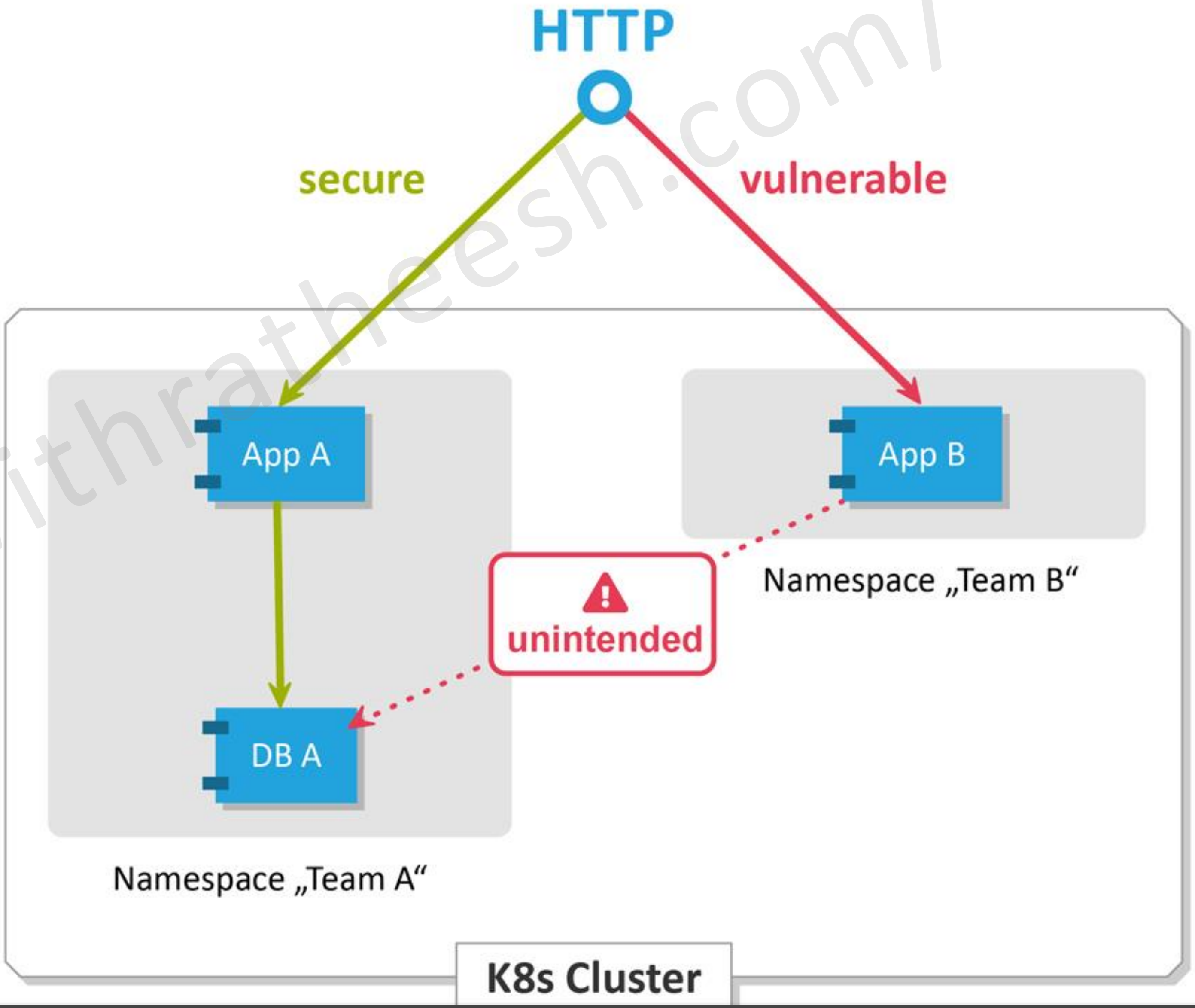


## Implementation with CNI

Not all CNIs support network policies. Mention popular CNIs like Calico, Weave, and Cilium that implement them.

## Monitoring and Logging

- Discuss tools and methods to monitor and log policy behaviour, ensuring policies act as intended.

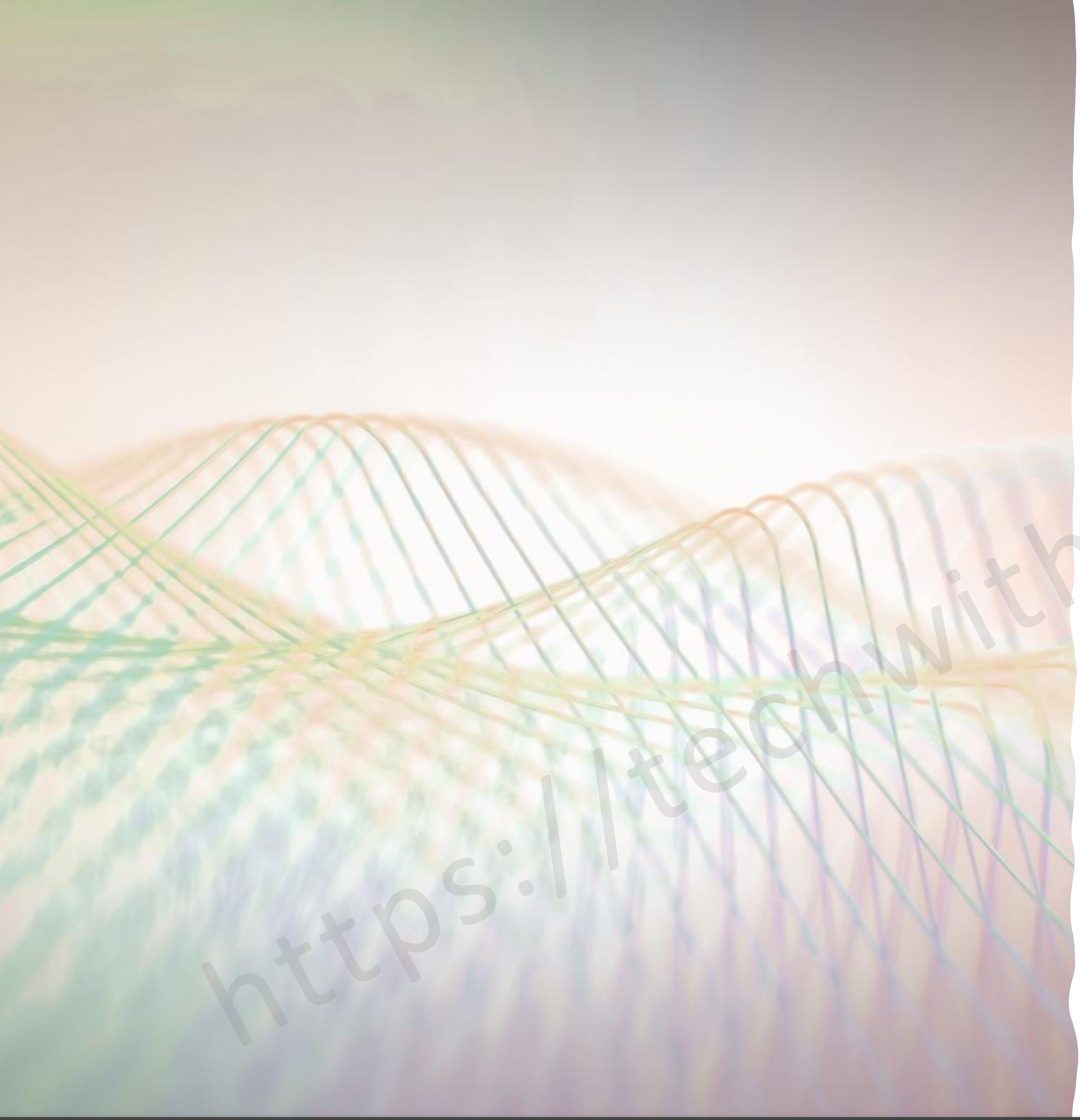


## Best Practices

Mention practices like default-deny policies, avoiding overly broad policies, and continuously auditing and refining network policies.

## Limitations

Despite their power, network policies have limitations. Discuss aspects like being pod-centric, relying on CNIs, and the lack of finer-grained controls.



## Conclusion

Reinforce the need for network policies in a Kubernetes environment and encourage continual learning as Kubernetes and its ecosystem evolve.

## Conclusion & Q&A

Importance of adopting network policies for secure K8s deployments- Open floor for questions

## Contact

+91 9446330906

RATHEESHKUMAR.2008@GMAIL.COM

www.techwithratheesh.com

linkedin.com/in/ratheesh-kumar-08722619

Thank You..!