Kubernetes

Part 3

Network Admission

- Refers to the traffic between
  - Pods in the cluster
  - External into the cluster
- Approaches to traffic control
  - Blacklist - default is to allow access unless its prohibited
  - Whitelist - default is to block access unless its permitted
- Possible approach to take for
  - Internal - trusted environment, blacklist
  - External - untrusted environment, whitelist

In Cluster Traffic

- Pods are not isolated when they are deployed
  - Default behaviour
  - Accept traffic from any sources - other pods and external entities
  - Connect to any destinations - other pods and external entities
- May want to restrict ingress and egress traffic
  - Eg. multi tenancy - only allow traffic for pods within a tenant but not to other tenants and visa versa
  - Eg. micro services - only allow pods that owns the database to connect to it
- Kubernetes uses network policy resource to manage traffic
  - Need to install a network plugin that supports network policy

Network Policy

- Network policy define a set of rules to regulate how traffic flow between a pod and its environment
- Network policy only has ‘allow’ rules, no deny rule
- Once a policy has been applied to a pod/set of pods
  - All traffic is blocked unless there is an allow rule
  - Eg cannot receive traffic from Ingress controller unless there is a rule that allows it - ingress rule
  - Eg. cannot query DNS unless there is a rule that allows it - egress rule
- Rules are additive
- Evaluated by ORing the rules, not AND
  - Evaluation are not affected by the order of the rules
  - Most firewalls process rules in the order which they are defined eg. iptables, ufw

Basic Network Policy

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: app-netpol
  namespace: myns
spec:
  podSelector:
    matchLabels:
      name: webapp
  policyTypes:
  - Ingress
  - Egress
  ingress:
    ...
  egress:
    ...
```

Network policy is a scoped resource. The rule applies to pod(s) in the specified namespace

Select all pods in the namespace that matches this selector

Policy types. Can be Ingress, Egress or both

Ingress and Egress rules

Rules

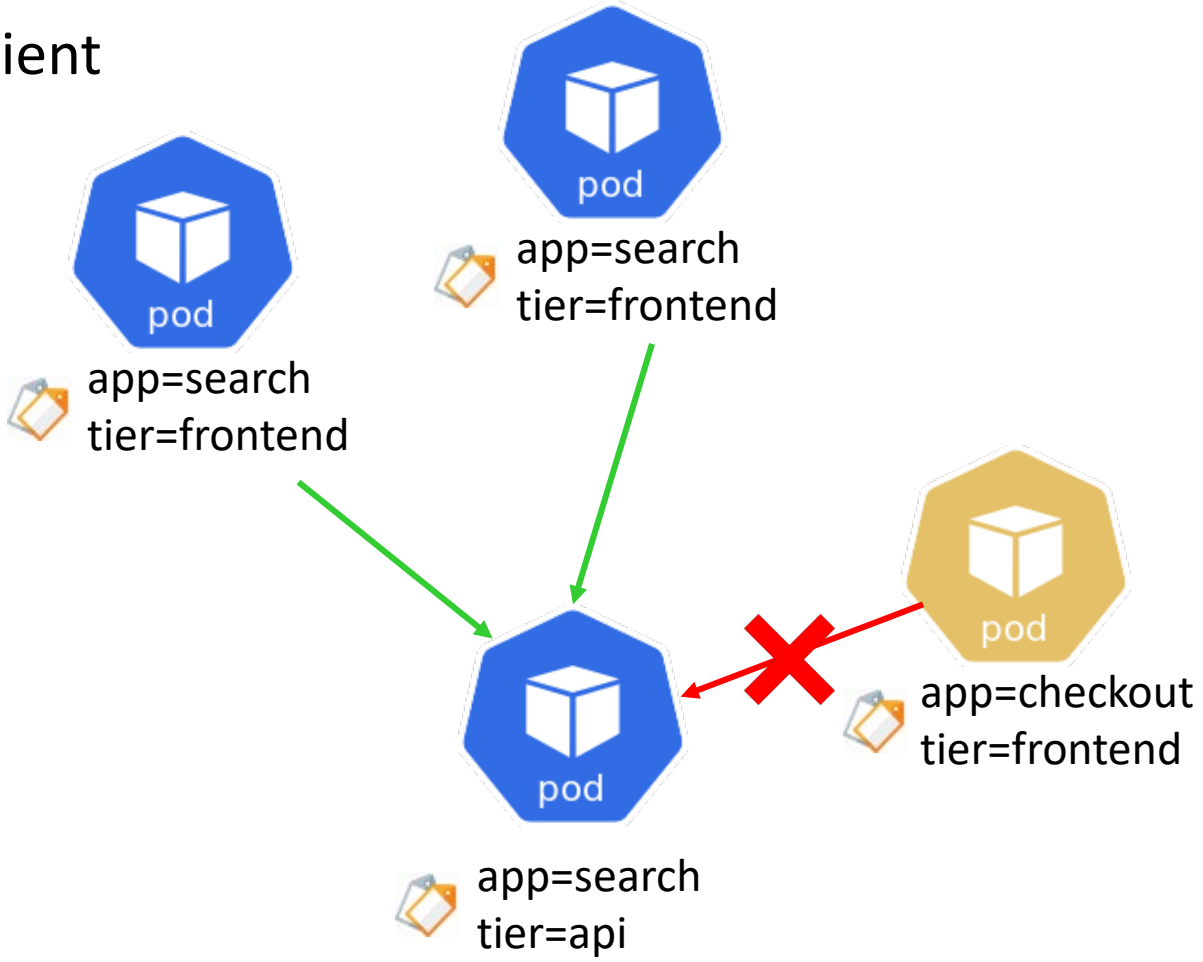
- Every ingress/egress rules has the following attributes
  - `from` - for Ingress
  - `to` - egress
  - `ports` - list of allowable ports to connect from or to
    - Allow connection from/to ports if ports is not specified

- Ingress/egress selectors
  - `podSelector`
    - Select pod(s) in the local namespaces
  - `namespaceSelector`
    - Select all pods in a given namespace
  - `namespaceSelector` and `podSelector`
    - Select specific pods from a given namespace
  - `ipBlock`
    - Select ingress/egress based on IP CIDR ranges

Network Policy Example 1

Allow only traffic from an application's client

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  ingress:
    from:
    - podSelector:
        matchLabels:
          app: search
          tier: frontend
```

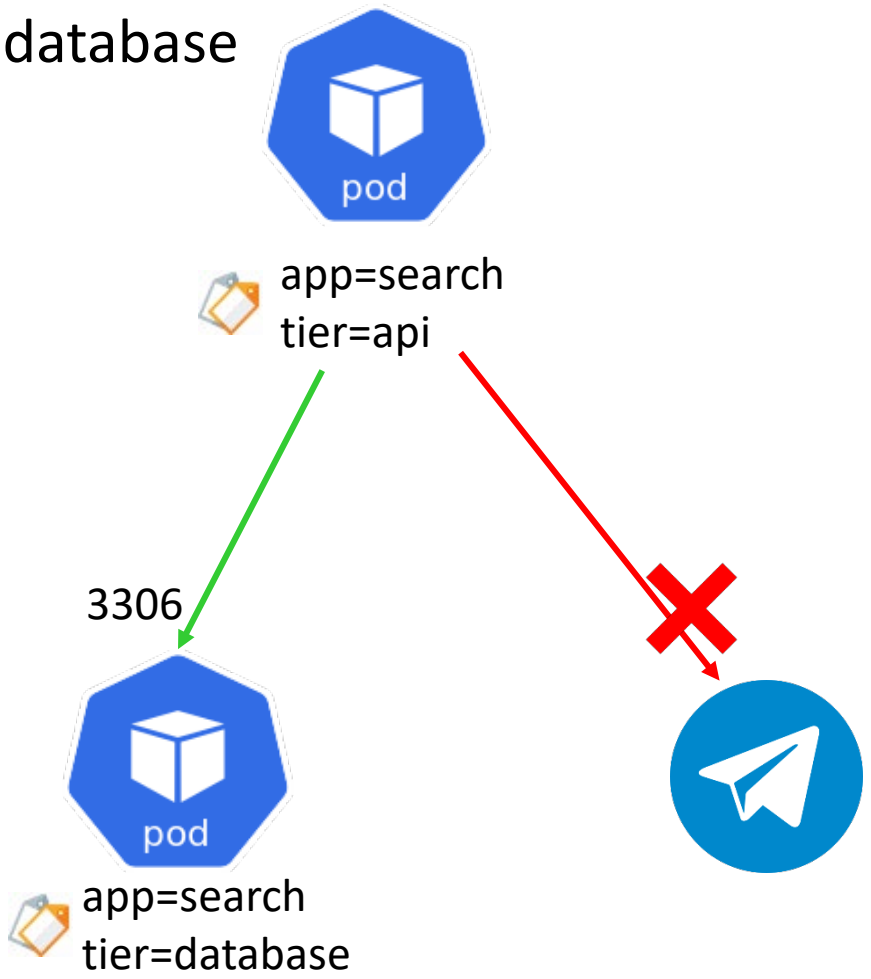


Assume pods are in `myns`

Network Policy Example 2

Allow outbound traffic only to the application's database

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Egress
  egress:
    to:
    - podSelector:
        matchLabels:
          app: search
          tier: database
    ports:
    - port: 3306
      protocol: TCP
```

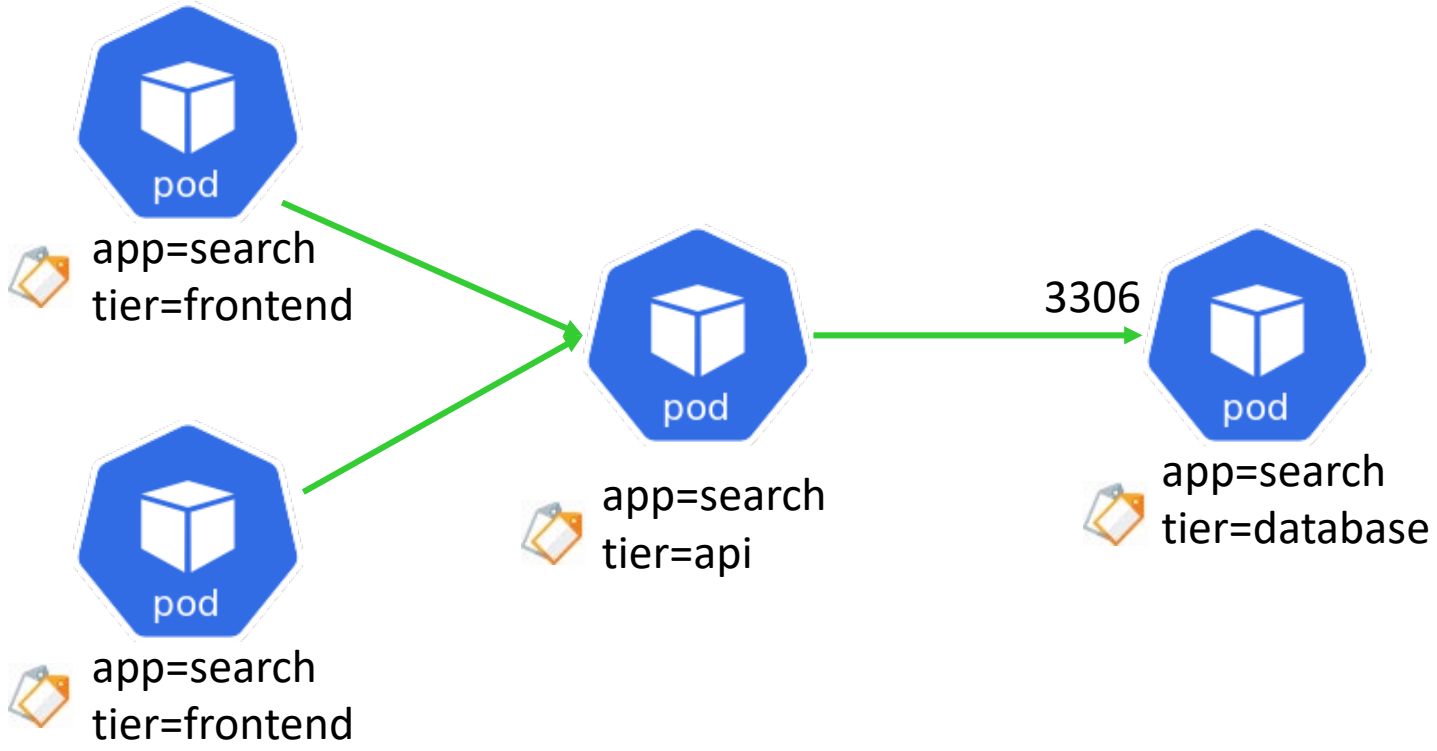


Assume pods are in `myns`

Network Policy Example 3

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  - Egress
  ingress:
    from:
    - podSelector:
        matchLabels:
          app: search
          tier: frontend
  egress:
    to:
    - podSelector:
        matchLabels:
          app: search
          tier: database
    ports:
    - port: 3306
      protocol: TCP
```

Allow traffic flow between all the pods in a micro service

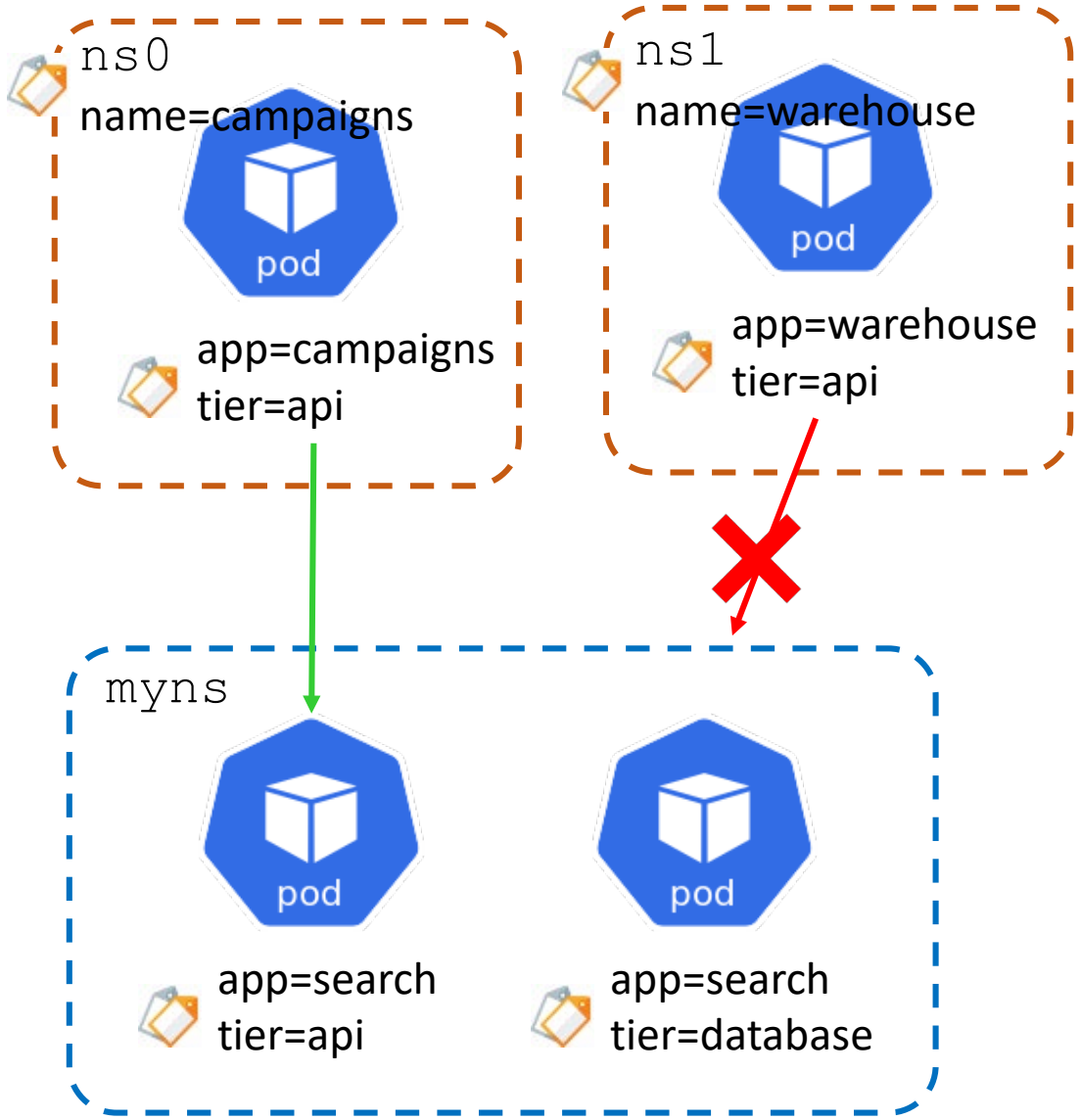


Assume pods are in `myns`

Network Policy Example 4

Allow traffic from pods of in all namespaces

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  ingress:
    from:
    - namespaceSelector:
        matchLabels:
          name: campaigns
```



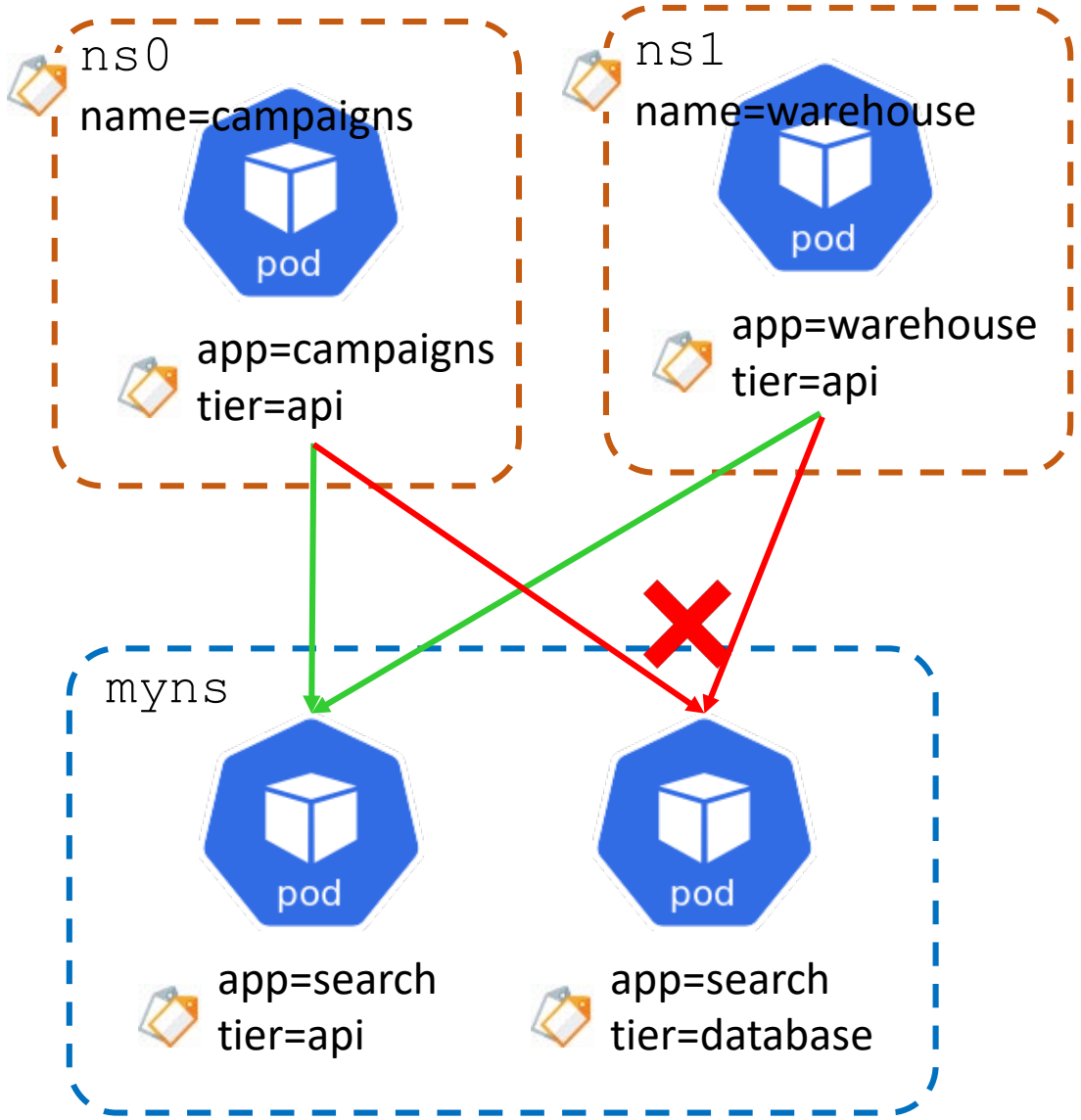
Assume pods are in `myns`

Network Policy Example 5

Allow traffic from pods of in all namespaces

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  ingress:
    from:
    - namespaceSelector: {}
```

Empty object matches any namespace



Assume pods are in `myns`

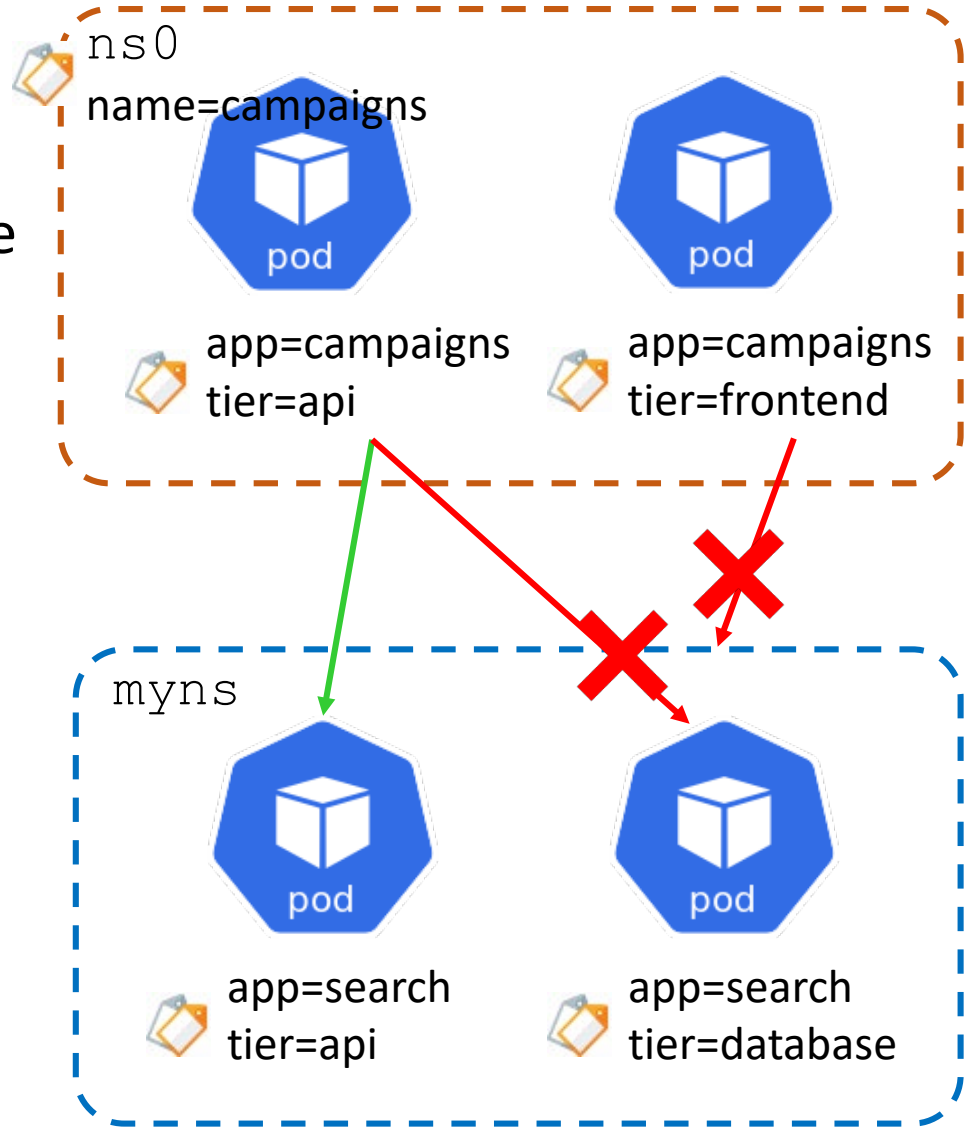
Network Policy Example 6

Allow traffic from certain pods of a namespace

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  ingress:
    from:
    - namespaceSelector:
        matchLabels:
          name: campaigns
      podSelector:
        matchLabels:
          tier: api
```

1 rule
2 conditions

AND



Assume pods are in `myns`

Network Policy Example 7

Allow all traffic from local and another namespace

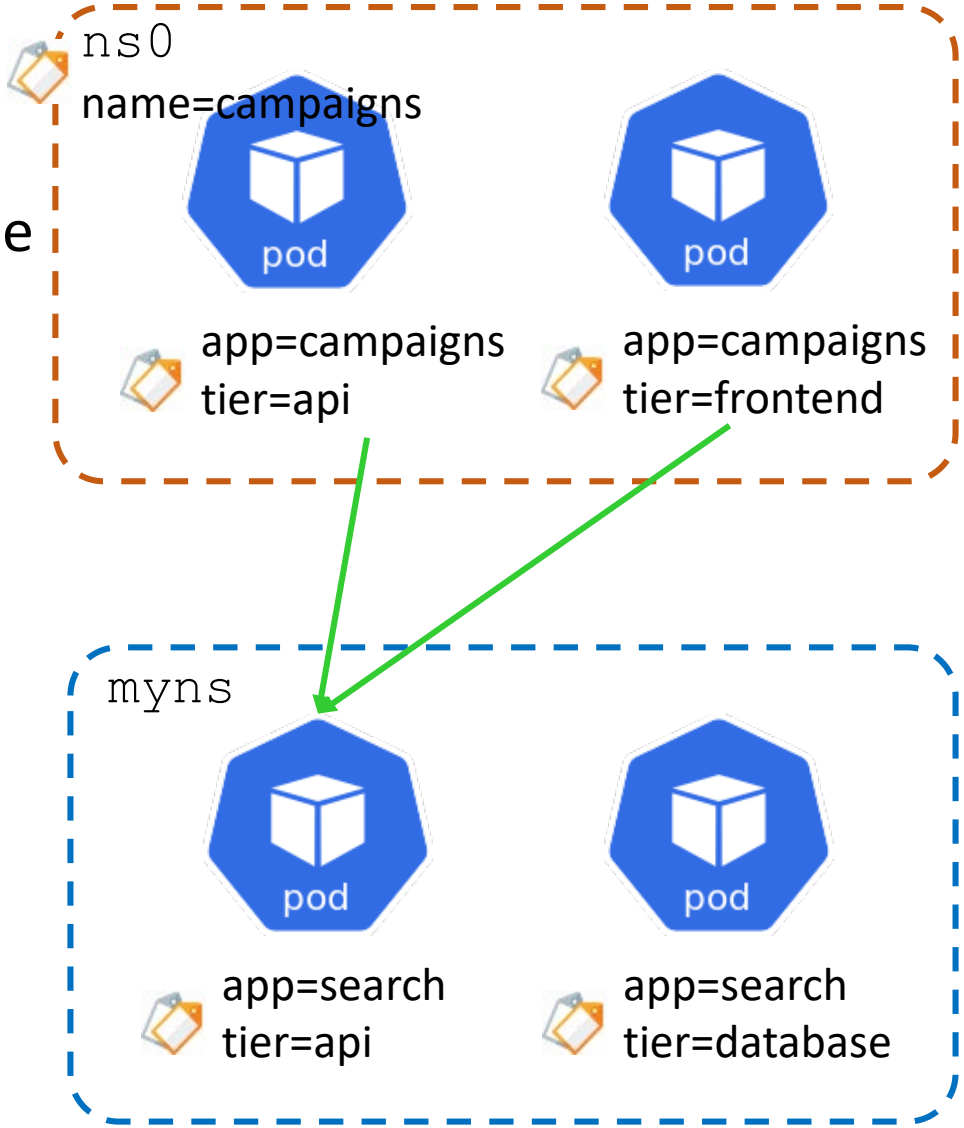
```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  ingress:
    from:
    - namespaceSelector:
        matchLabels:
          name: campaigns
    - podSelector:
        matchLabels:
          tier: api
```

2 rules
1 condition

OR

Any pods from `campaigns` namespace

Pods from local namespace

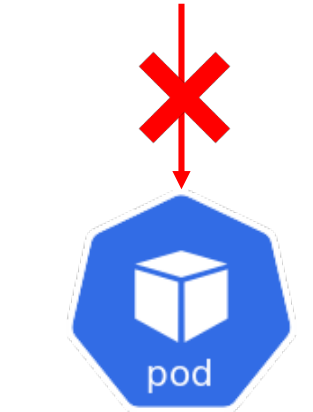


Assume pods are in `myns`

Network Policy Example 8

Deny all traffic

```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  - Egress
```



app=search
tier=api

Assume pods are in `myns`

Network Policy Example 8

Allow all traffic from any source and to any destination - default behaviour

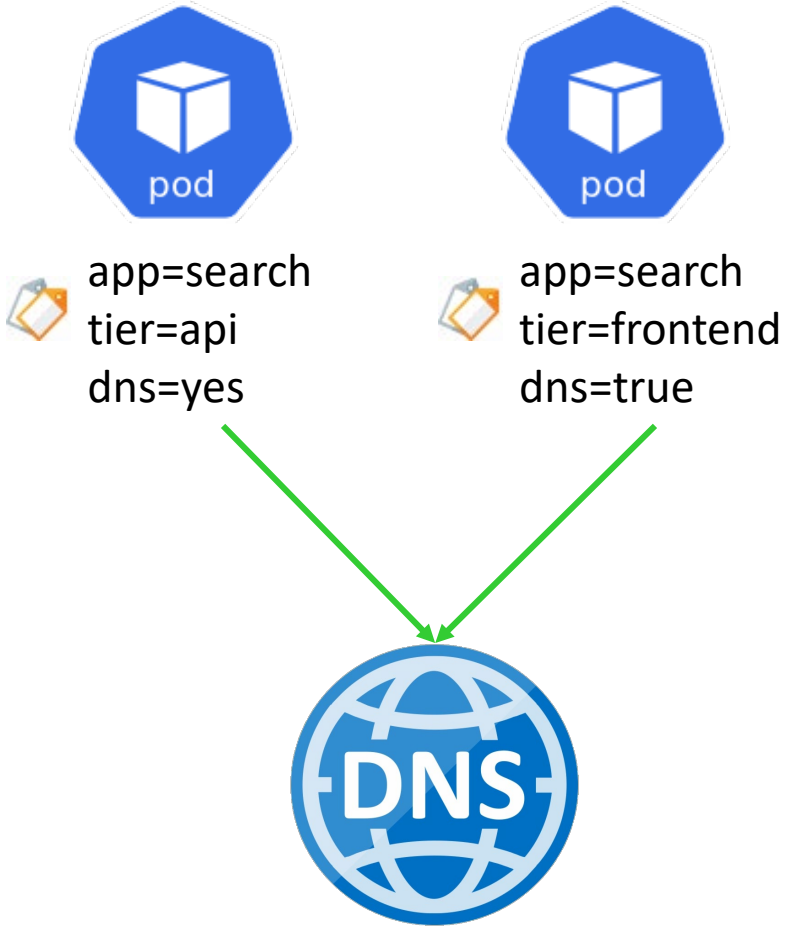
```
spec:
  podSelector:
    matchLabels:
      app: search
      tier: api
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - {}
  egress:
  - {}
```

Assume pods are in `myns`

Network Policy Example 9

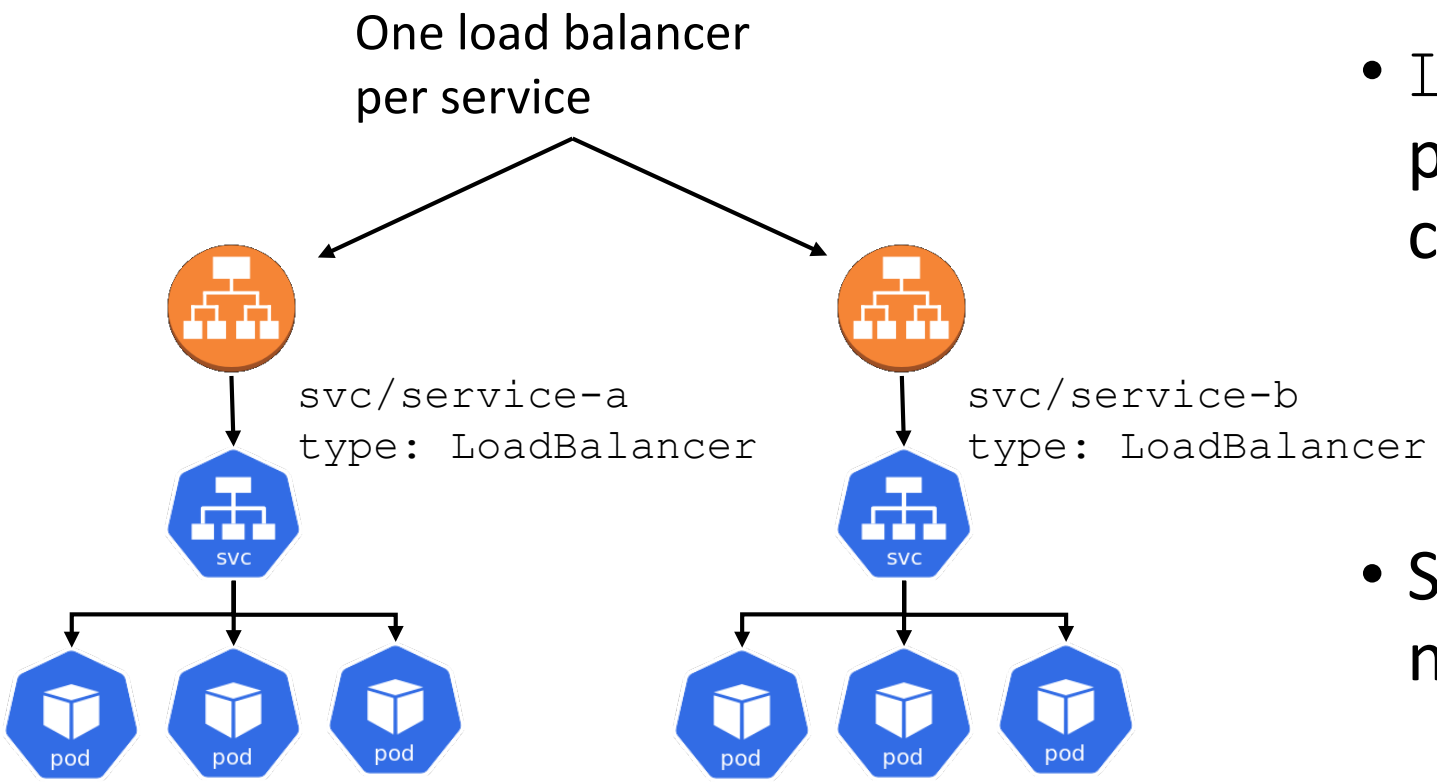
Allow pods to query DNS

```
spec:
  podSelector:
    matchExpressions:
    - key: dns
      operator: In
      values: [ “yes”, “true” ]
  policyTypes:
  - Egress
  egress:
  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
```



Assume pods are in `myns`

`LoadBalancer` Service Type

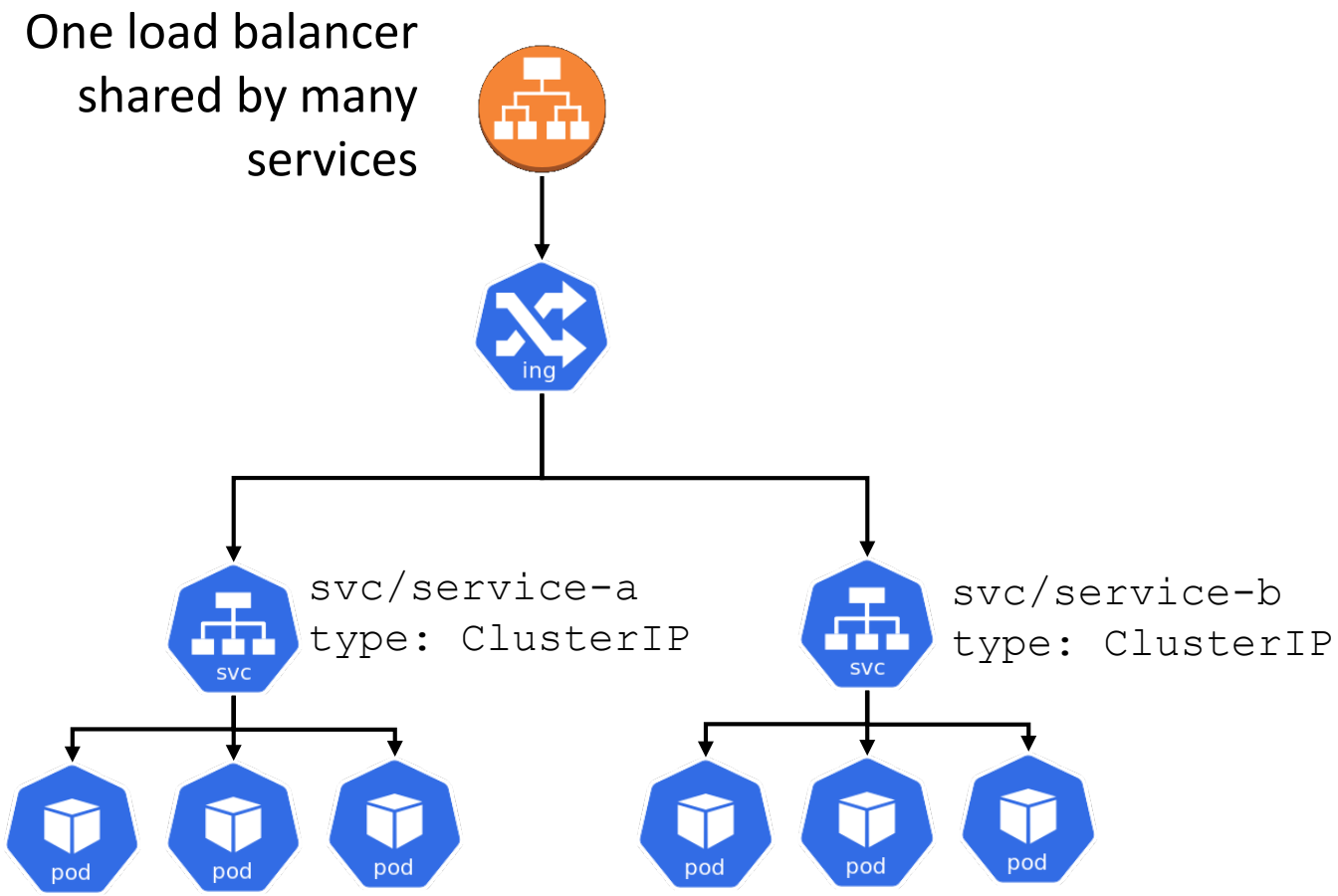


- `LoadBalancer` service provision a load balancer on cloud provider
  - Allows external traffic into the cluster
  - 1 to 1 correlation
- Smaller services may not need a load balancer
  - No way of sharing
  - Not cost effective

Single Entry Point

- Not secure if there are multiple uncontrolled entry points into the cluster
  - This is the case if we allow LoadBalancer service type
- Should have a single entry point
  - A choke point where all traffic enters
- Implement and enforce enterprise level policies at a single point
  - Eg. security, logging, etc
  - Certificates
- Reduce attack surface

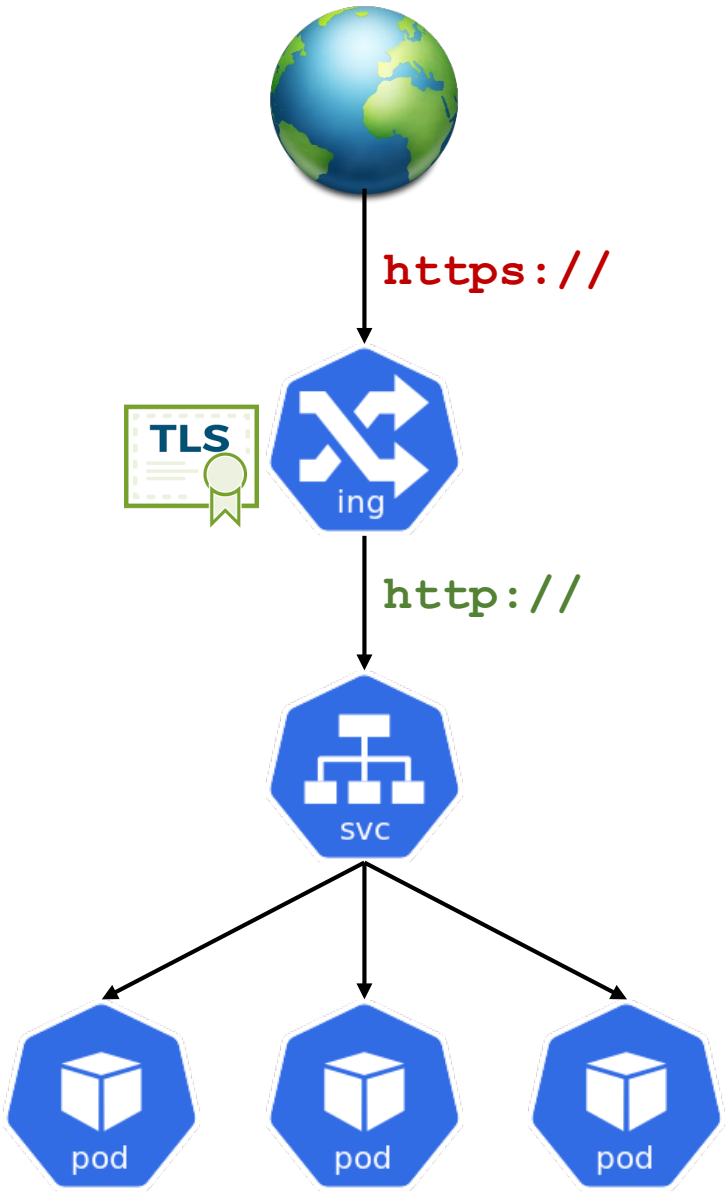
Ingress



- Like `LoadBalancer` service type but is shared between many `ClusterIP` type service
  - Regulate external traffic to many different services
  - Traffic can be routed based on FQDN, resource or both
- Different ingress controllers have additional capabilities eg. OpenAPI, authentication, tracing, etc.
  - Implementation dependent

TLS

- Securing and endpoint with TLS provides the following benefits
  - Encrypt the communication channel between the client and the application
    - Protect sensitive data from being snoop or sniffed
  - Ensure that the client is talking to the 'correct' application
    - Can verify the server from its certificate
- TLS should be enabled at ingresses and not at individual application/pod
  - Better certificate and key management
- TLS terminates at the ingress
  - Traffic between the ingress and the pod is not encrypted



Enable TLS on Ingress

Example only. Generate a X.509 certificate for app.acme.com domain

```
openssl req -x509 \
  -key app.key \
  -out app.cert \
  -days 1000
  -subj "/CN=app.acme.com"

kubectl create secret tls app-tls \
  --key=app.key \
  --cert=app.cert \
  -n myns
```

Create a TLS secret to be used by the ingress resource

```
apiVersion: networking.k8s.io/v1
kind: Ingress
...
spec:
  ingressClassName: nginx
  rules:
  - host: app.acme.com
    http:
     paths:
       ...
  tls:
  - hosts:
    - app.acme.com
    secretName: app-tls
```

Certificate to use for the host. Domain name must match the name in the certificate