Kubernetes 103

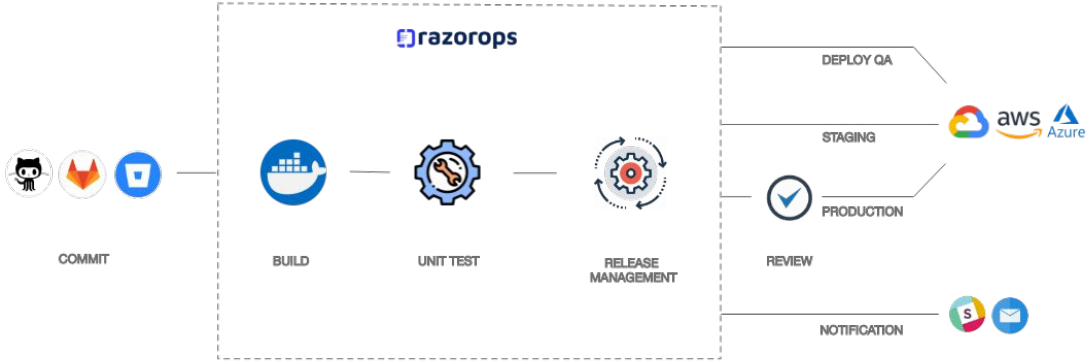
Service Discovery, Networking and RBAC

Dinesh Yadav (@dineshoi)

Sponser

Razorops

Fast, modern CI/CD system for software teams to test, build and ship quickly.



https://razorops.com
@razorops

Prerequisite

- A working k8s cluster
- Kubectl
- Docker basics
- Talk 101 & 102
- Kubernetes Pod, deployment, Service basics

TOC

- Service, Networking and Discovery
- Ingress
- Authorization (RBAC)
- Authentication (Static, Token, Certs, SSO)

Find source code on github

Service Discovery & Networking

Topics to cover -

- How to communicate inside and outside of cluster
- DNS for services and pods
- Connecting apps
- Service Kinds
- Ingress

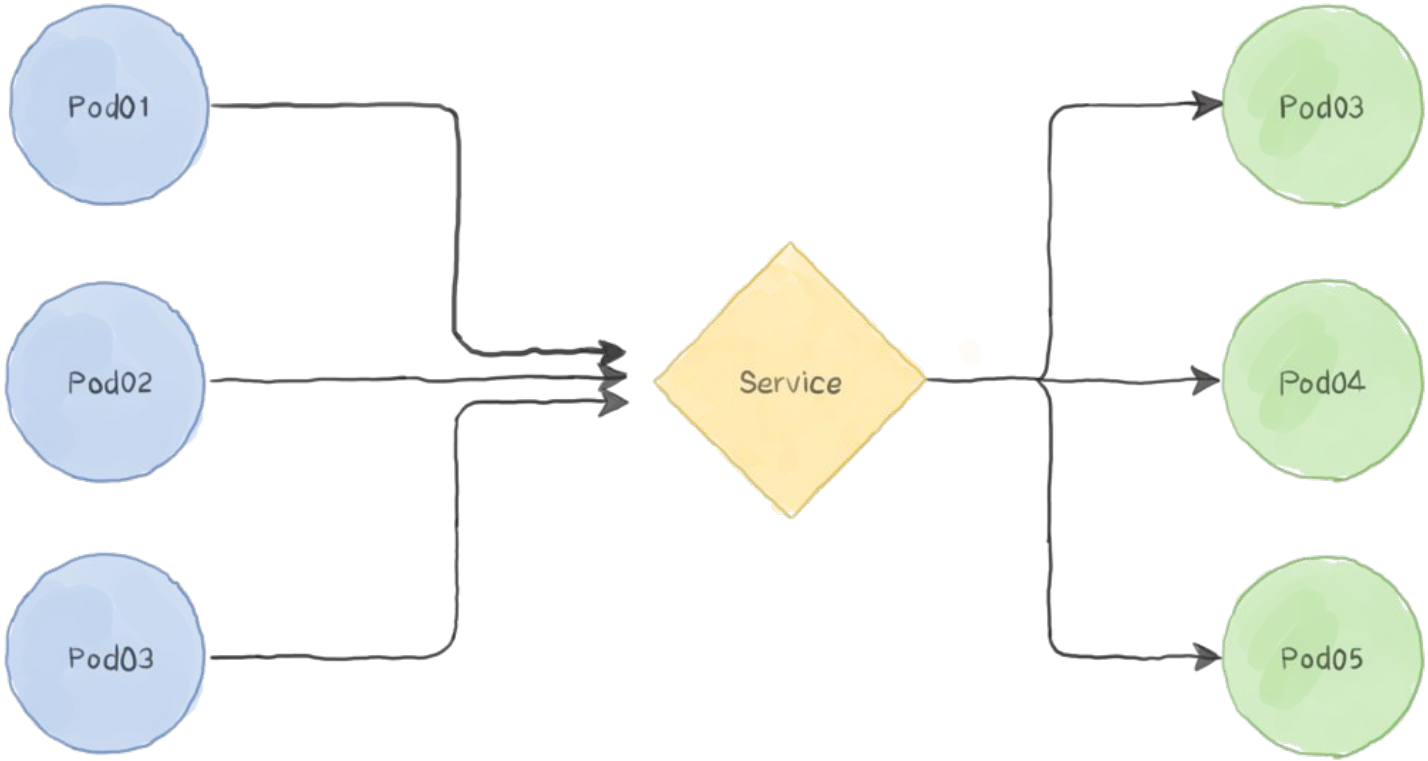
InCluster Communication

- Pods are ephemeral, so does PodIPs
- Acts as a load balancer for pods inside cluster
- Use labels and selectors
- FQDN ($svc.$namespace.svc)
- Creates iptables rules (kube-proxy)
- ClusterIP and Headless kind

```yaml
apiVersion: v1
kind: Service
metadata:
  name: thesvc
spec:
  ports:
    - port: 80
      targetPort: 9876
  selector:
    app: sise
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rcsise
spec:
  selector:
    app: sise
  template:
    metadata:
      name: somename
      labels:
        app: sise
    spec:
      ...
```

Service acts as a Loadbalancer within cluster



Exposing Services

Type: NodePort

- Exposes a port on every Node

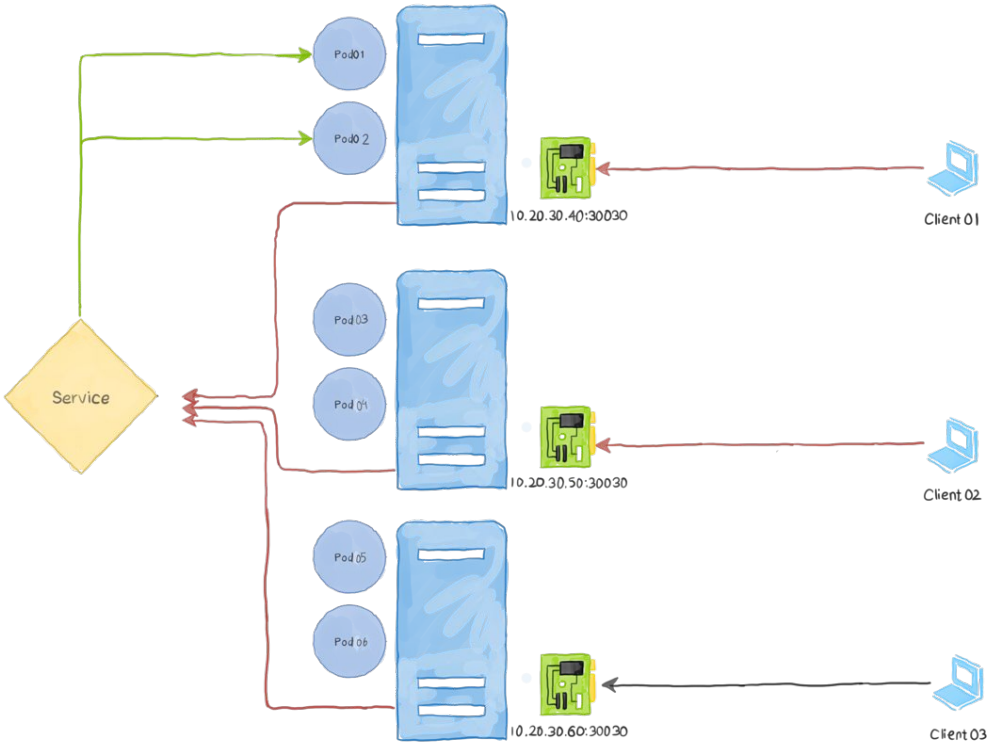
```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    name: mysql
spec:
  type: NodePort
  ports:
    - port: 3036
      nodePort: 30036
      name: http
  selector:
    name: mysql
```

Type: LoadBalancer

- Manages external access to the services
- serves one service
- only works in cloud environment

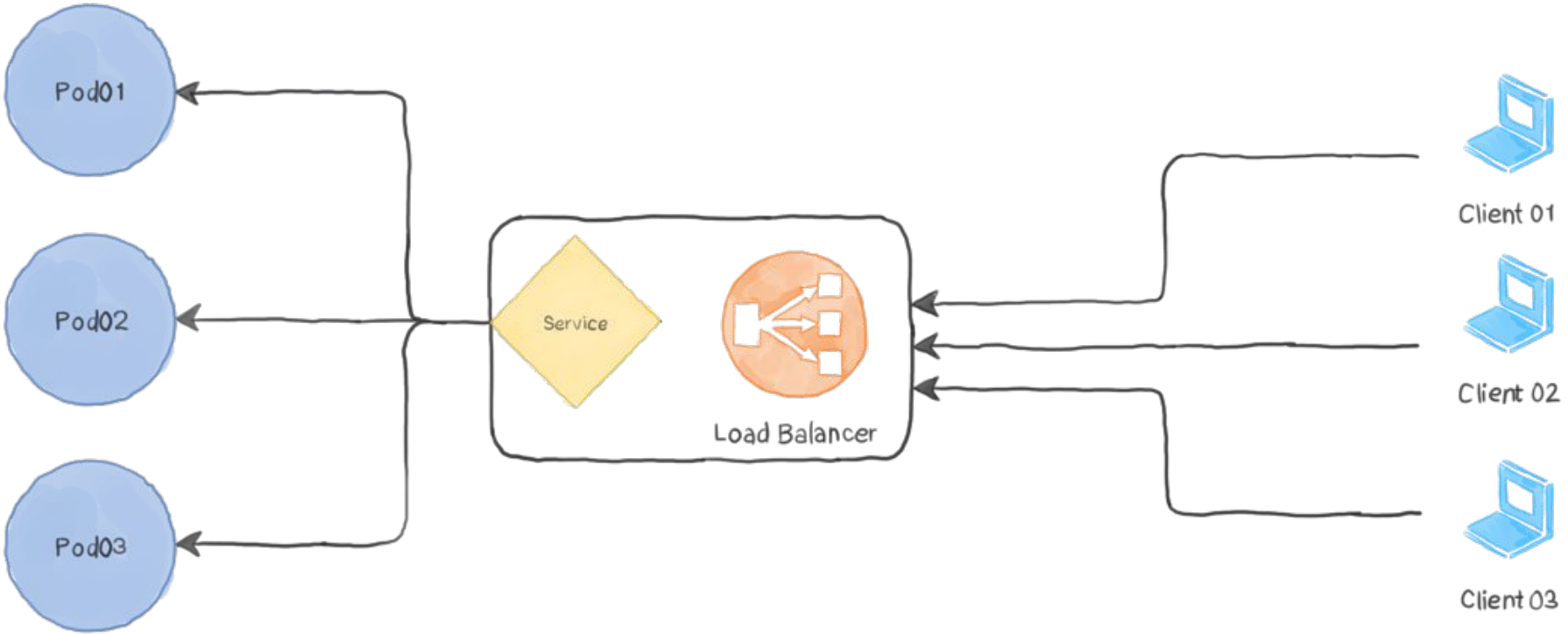
```yaml
apiVersion: v1
kind: Service
metadata:
  name: thesvc
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 9876
  selector:
    app: sise
```

NodePort service exposes a random port on every node



Type: Loadbalancer

- Service is exposed with a loadbalancer provisioned by a cloud
- Only works on cloud platform

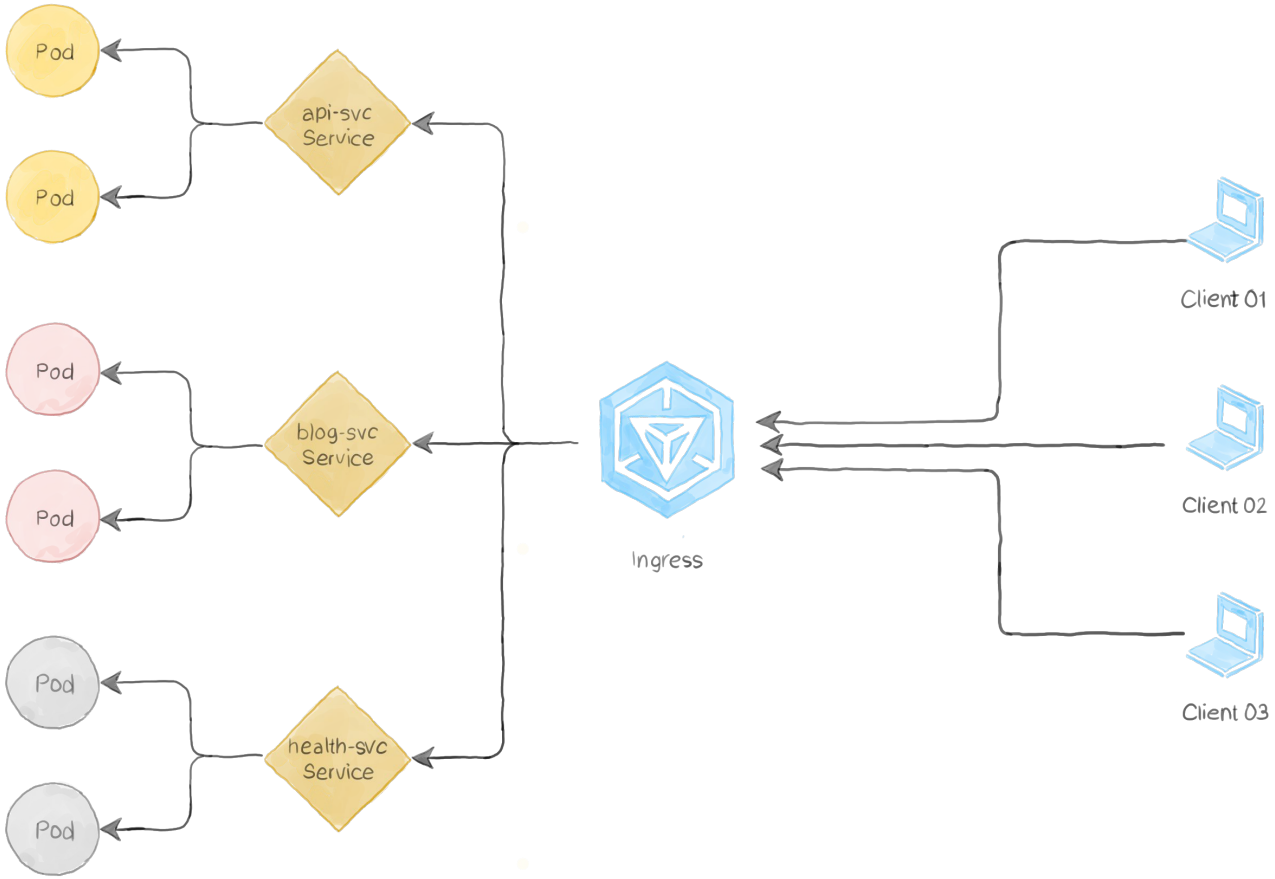


Ingress

- Manages external access to the services
- serves multiple services
- Needs ingress controller running
- can terminate SSL, have routing based on path and hostname

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          serviceName: test
          servicePort: 80
```

- Exposing multiple services via an Ingress
- Needs an ingress controller



Service Types

Common Name	Configuration	Use Case
Internal	type: ClusterIP	You need to access pods internally
NodePort	type: NodePort	You need to accept incoming connections and you have a node
LoadBalancer	Type: LoadBalancer	You need to accept incoming connections and you have multiple nodes
Headless	clusterIP: None	Used for Statefulsets or load balancing on the client-side
Ingress	A separate k8s resource	One external endpoint for multiple services

Role Based Control Access

Kubernetes RBAC

- Authorization layer to restrict unwanted access
- Use objects and HTTP verbs to define access boundaries
- Adds security to a Kubernetes cluster
- Represented in declarative form using resources

Role/ClusterRole

- Defines permission on K8S Resources using CURD operations
- Role is a namespaced, but ClusterRole is not
- Verbs = list, create, get, watch, patch, update
- Resource - any standard object or custom CRD

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: get-pods
rules:
 - apiGroups: ["*"]
   resources: ["pods"]
   verbs: ["list"]
```

RoleBinding/ClusterRoleBinding

- Binds *Subject(s)* with a *Role*
- RoleBinding is namespaced, but ClusterRoleBinding is not
- *Subject -* An identity for humans and processes accessing the cluster.
- *roleRef* refers to a Role or ClusterRole.

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: get-pods-rb
subjects:
- apiGroup: ""
  kind: User
  name: viewpod
roleRef:
  apiGroup: ""
  kind: Role
  name: get-pods
```

Generating credentials

- ServiceAccount can be used with RBAC
- ServiceAccount Token acts as Bearer Token to k8s API
- Good for processes accessing cluster
- Generate kubeconfig for ServiceAccount from script

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: get-pods-rb
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: ServiceAccount
  name: viewsa
roleRef:
  apiGroup: ""
  kind: Role
  name: get-pods
```

Next Session

Managing Kubernetes manifests
(using Kustomize and Tanka)

What can we can do for you ?

We are always available to talk about your challenges moving forward microservices and containers.

Request a call here.