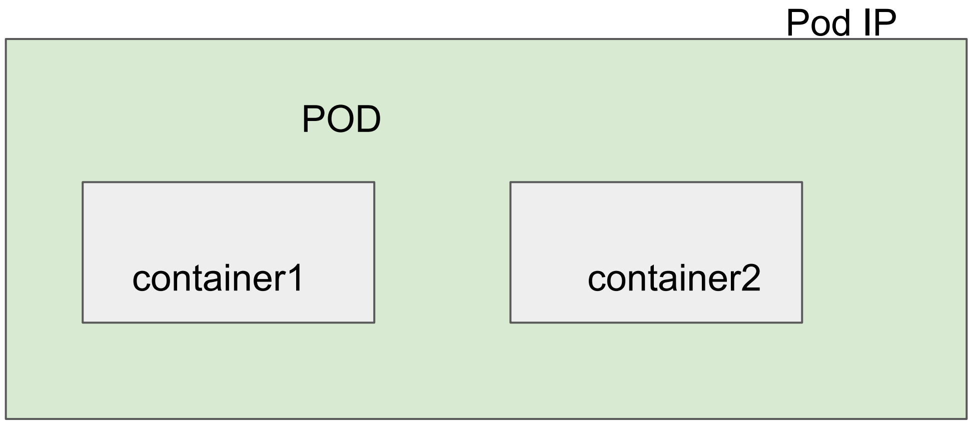
kubernetes

Resources

- Pods
- ReplicasSet
- Deployment
- Services

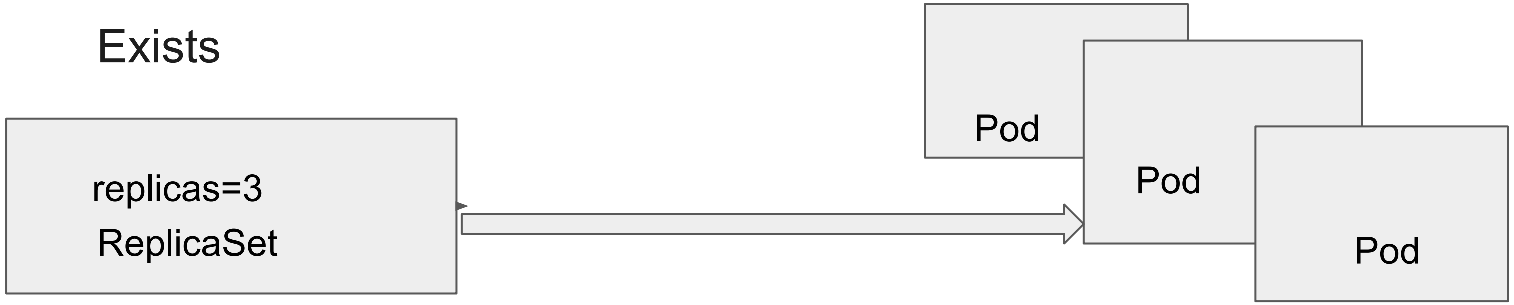
#Pods

- An environment to run containers
- It have network stack, kernel namespaces and one or more container running
- Container always runs inside a pod
- Pod can have multiple containers
- It is unit of scaling in k8s



#ReplicaSet

- A replica set makes sure the specified replicas of pods are always running.
- ReplicaSets manages all the pods with labels that match the selector.
- There are three kinds of operator we can use with ReplicaSets:
  - In
  - Notin
  - Exists

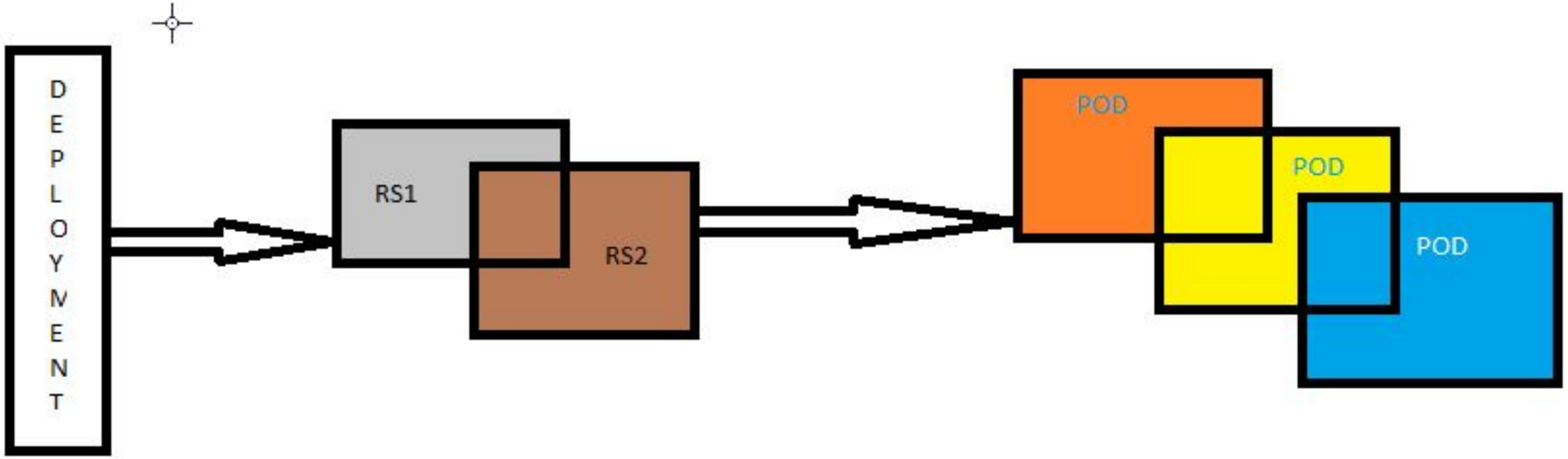


#Deployment

- A `Deployment` owns and manages one or more `ReplicaSets`. And `Replica Set` manages the basic units in Kubernetes - `Pods`.

- Why Deployment manages multiple ReplicaSets?

  The answer is Kubernetes wants to support `rollback` mechanism. Kubernetes creates a new ReplicaSet each time after the new Deployment config is deployed and also keeps the old ReplicaSet. So that we can rollback to the previous state with old ReplicaSet.


DEPLOYMENT
RS1
RS2
POD
POD
POD

#Rolling Update

In order to support rolling update, we need to configure the update strategy first.

```
minReadySeconds: 5
strategy:
  type:RollingUpdate
  rollingUpdate:
    maxSurge:1
    maxUnavailable:1
```

- minReadySeconds: Kubernetes waits specific time til the next pod creation.
- maxSurge: amount of pods more than the desired number of Pods.
- maxUnavailable: amount of pods that can be unavailable during the update process.

```
kubectl apply -f nginx.yaml --record
# format
kubectl set image deployment <deployment> <container>=<image>
# example
kubectl set image deployment nginx nginx=nginx:1.11.5 --record
# newer image version image: nginx:1.11.5
kubectl replace -f new-nginx.yaml --record
kubectl edit deployment nginx --record
kubectl rollout status deployment nginx
kubectl rollout pause deployment <deployment>
kubectl rollout resume deployment <deployment>
kubectl rollout history deployment nginx
# to previous revision
kubectl rollout undo deployment <deployment>
# to specific revision
kubectl rollout undo deployment <deployment> --to-revision=<revision>
# example
kubectl rollout undo deployment nginx --to-revision=1
```

#Services

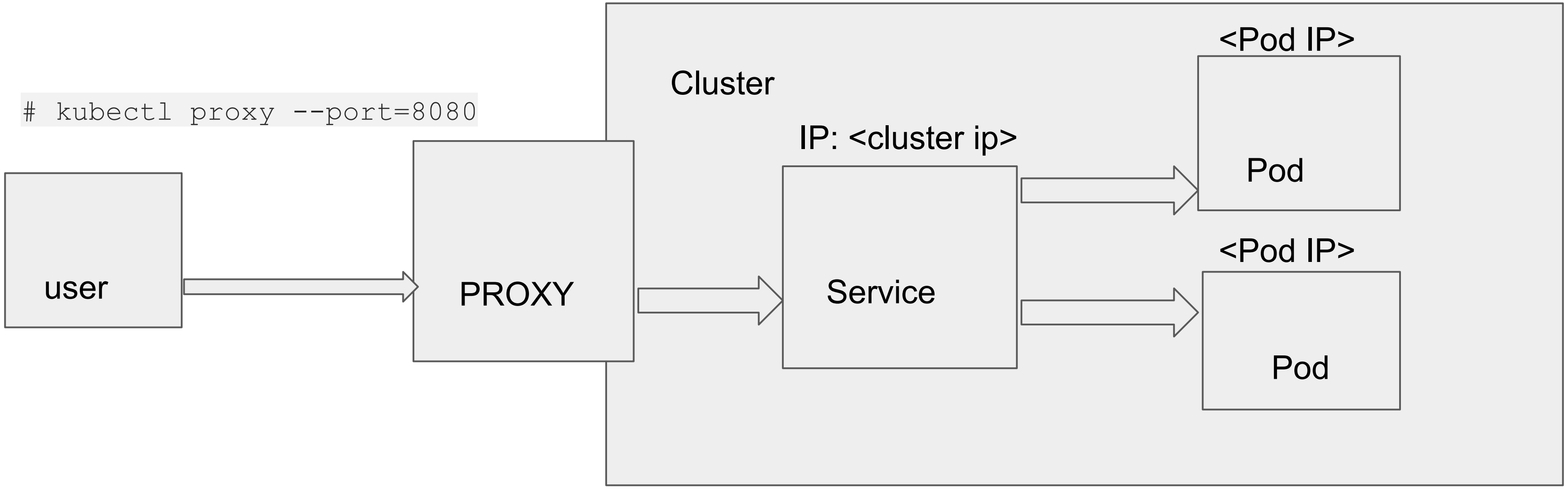
Pods comes and go with different IPs. To distribute load and act as a single source of interaction to all pods of an application, **service** play the role.

- Has single IP and DNS
- Created with a manifest JSON file
- All new pods gets added/registered to the service
- Which pod should be assigned to which services is decided by **labels**
- **service** and **pods** have **labels** on the basis of which service identifies its **pods**
- only sends traffic to healthy pods
- uses tcp by default (udp is also supported)

#ClusterIP

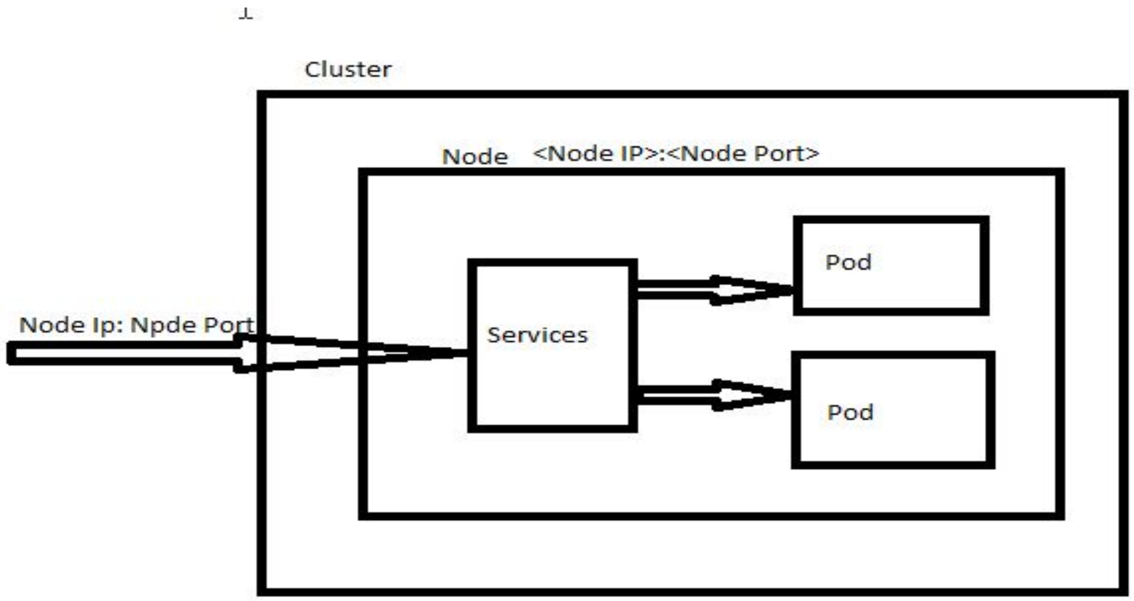
- A ClusterIP service is the default Kubernetes service. It gives you a service inside your cluster that other apps inside your cluster can access. There is no external access.

```
# kubectl proxy --port=8080
```



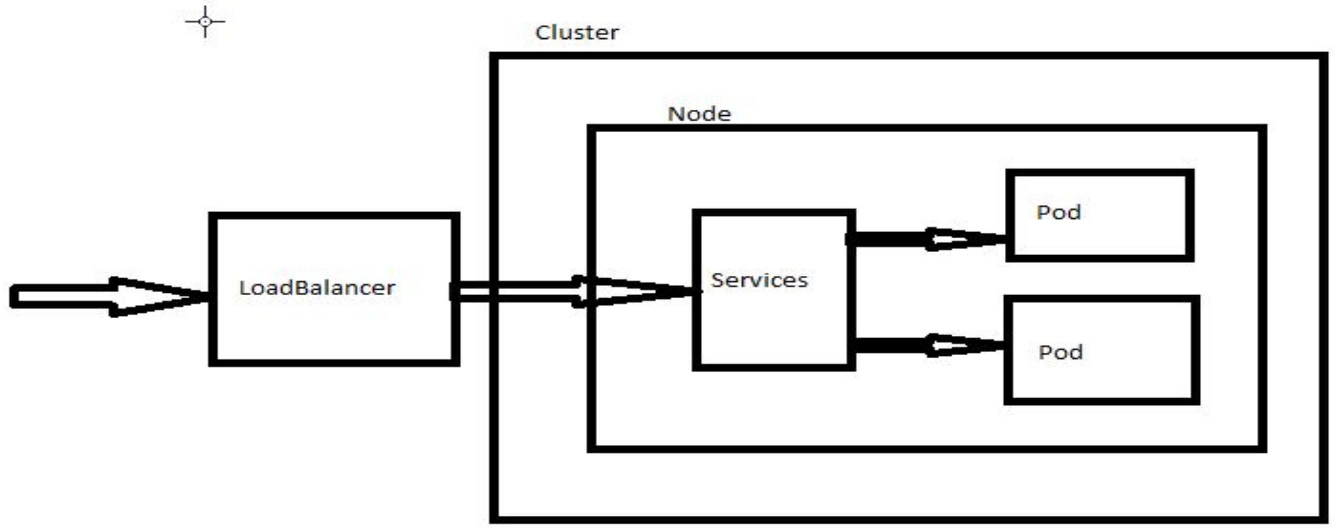
#NodePort

- A NodePort service is the most primitive way to get external traffic directly to your service. NodePort, as the name implies, opens a specific port on all the Nodes (the VMs), and any traffic that is sent to this port is forwarded to the service
- Port <30000-32767>



#LoadBalancer

- A LoadBalancer service is the standard way to expose a service to the internet. On GKE, this will spin up a Network Load Balancer that will give you a single IP address that will forward all traffic to your service.

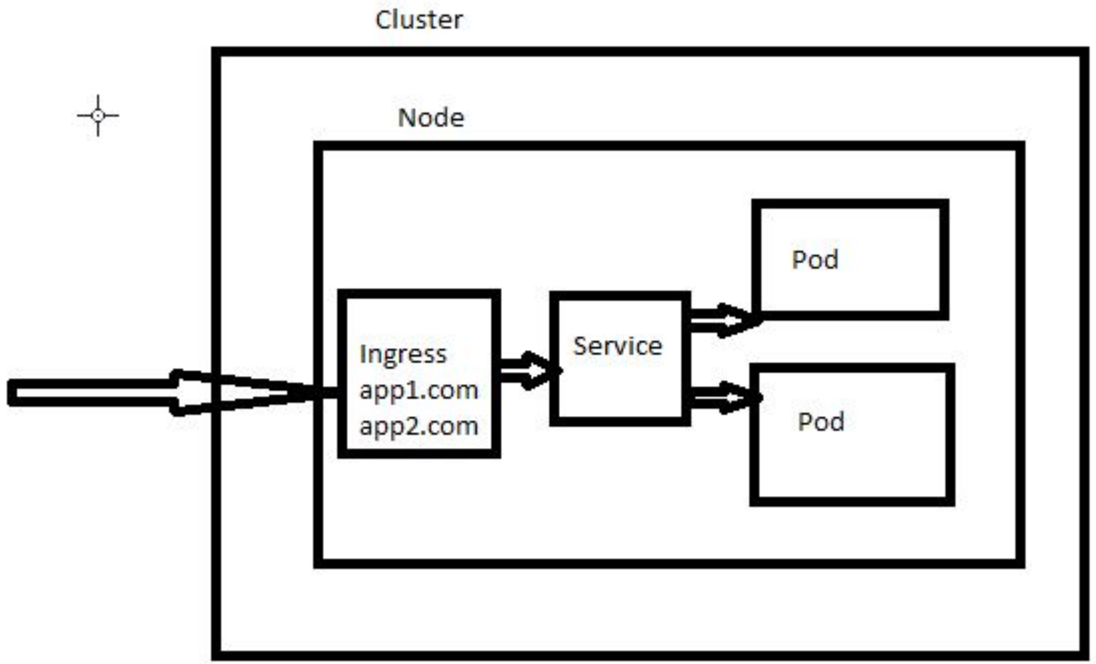


#Ingress

- Unlike all the above examples, Ingress is actually NOT a type of service.

  Instead, it sits in front of multiple services and act as a “smart router” or

  entrypoint into your cluster.



#K8s Dashboard:

- To create k8s dashboard we need to create some resource, run the below command to create the resource.

  kubectl apply -f
  https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta8/aio/deploy/recommended.yaml
- To access the dashboard need service account with full admin role to make any changes from dashboard. Copy the below content in yaml file and create with kubelet.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eks-admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: eks-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: eks-admin
  namespace: kube-system
```

Command to get token:

```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')
```

URL to access dashboard:

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/login

THANK
YOU