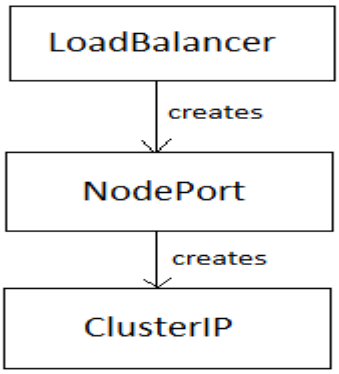
Kubernetes Services

- Kubernetes service is an object just like PODs, Replica Sets or Deployments.
- One of its use case is to listen to a port on the node and forward requests on that port to a port on the pod running the application.
- There is three type of services:
    1. NodePort
    2. ClusterIP
    3. LoadBalancer



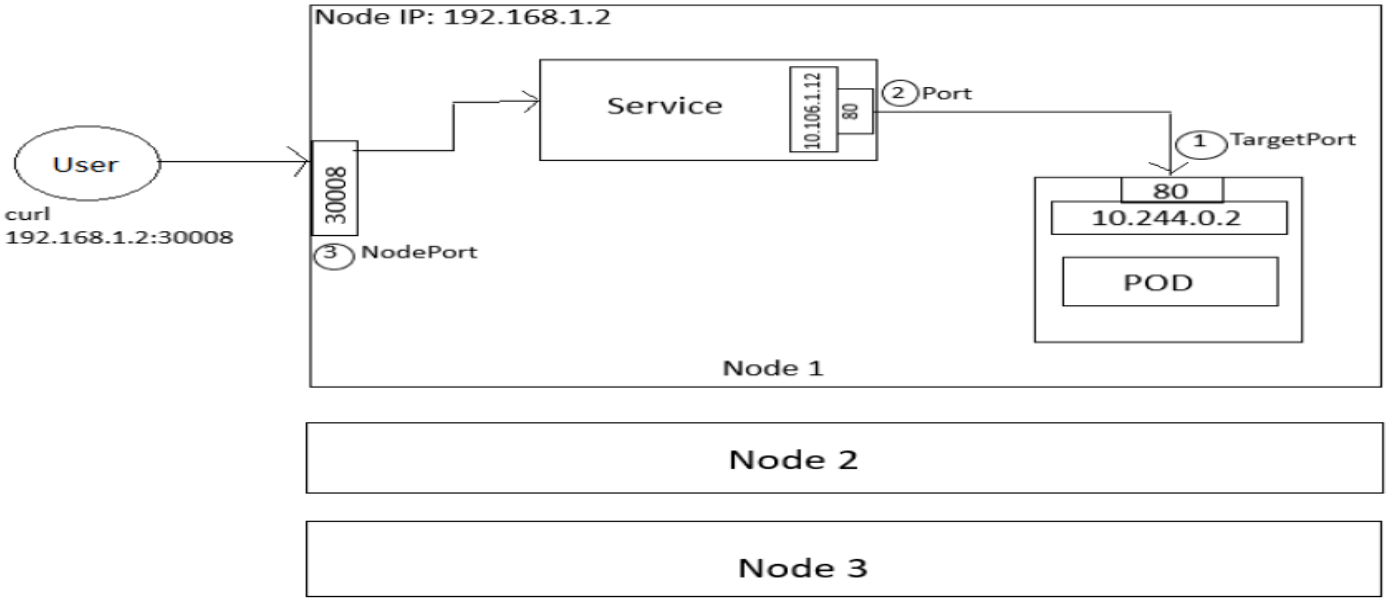
If you create a NodePort service it also creates a ClusterIP one. And if you create a LoadBalancer it creates a NodePort which then creates a ClusterIP.

1) NordPort: -

NordPort were the service makes an internal pod accessible on a port on the node. (This service listens to a port on the node and forward request to pods)

NodePort port Range is 30000 to 32767

I) When you have a single pod on a single node

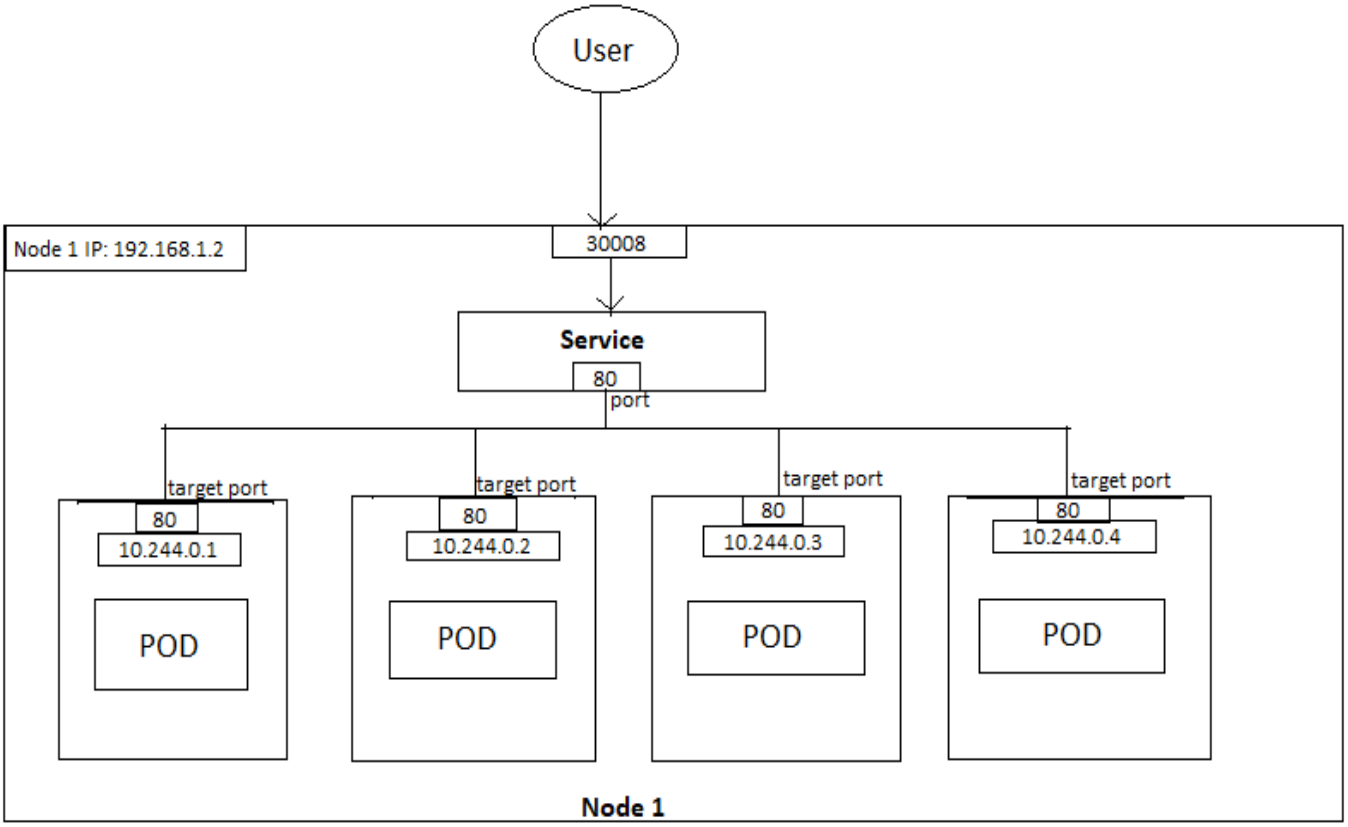


The port 30008 on the node itself which we used to access the web server externally

service-definition.yml

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  type: NodePort
  ports:
  - targetPort: 80
    port: 80
    nodeport: 30008
    protocol: TCP
  selector:
    app: myapp
    type: front-end
```

II) When you have a multiple pod on a single node (In production environment)

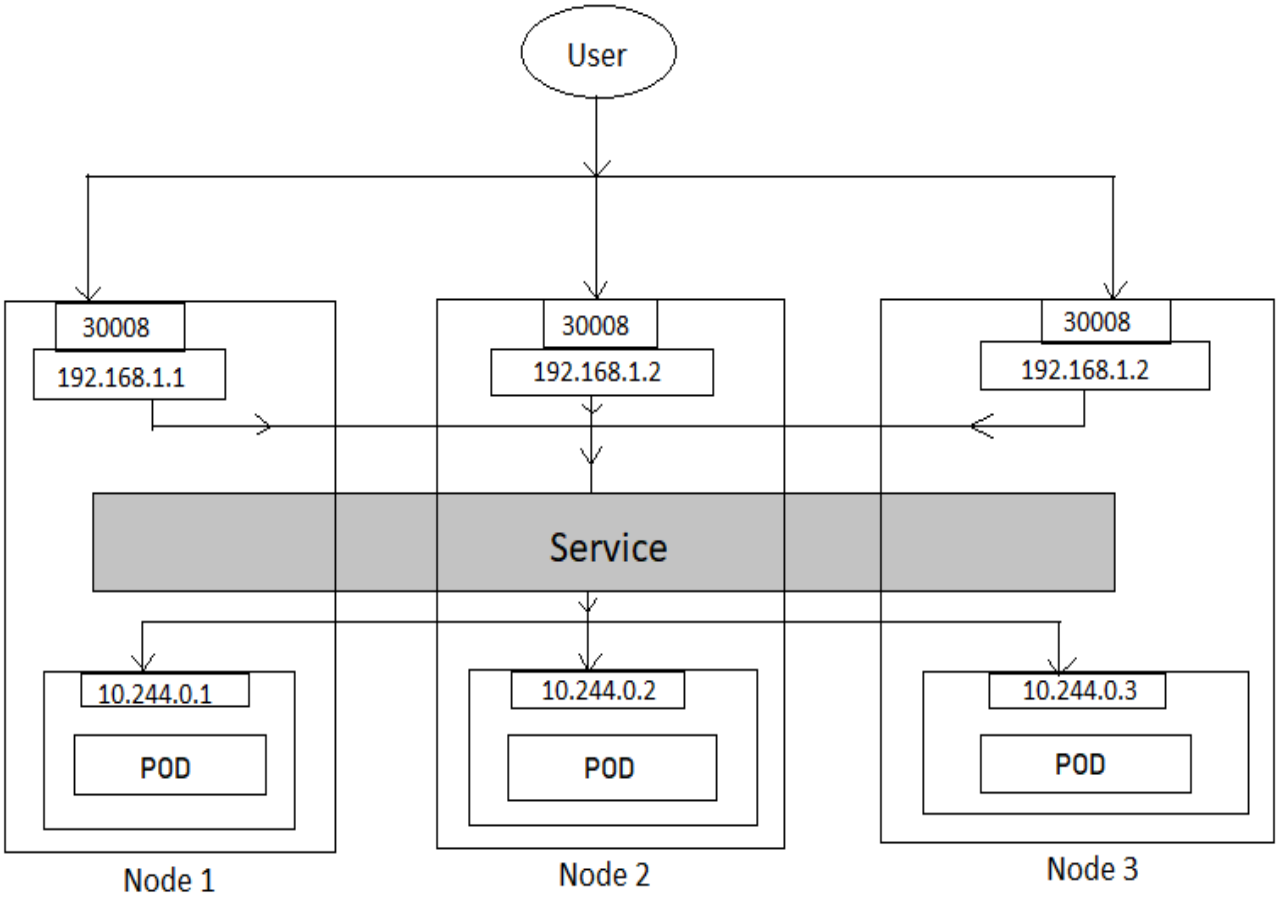


We have multiple similar pods running our web application and they all have the same labels and its value.

The same label is used to as a selector during the creation of the service. So, when the service is created is looks for a matching pod with the label and finds 3 of them. Then the service automatically selects all the 3 pods as endpoints to forward the external requests coming from the user.

It uses Random Algorithm to balance the load across the 3 different pods.

III) When the multiple pods are distributed across the multiple nodes

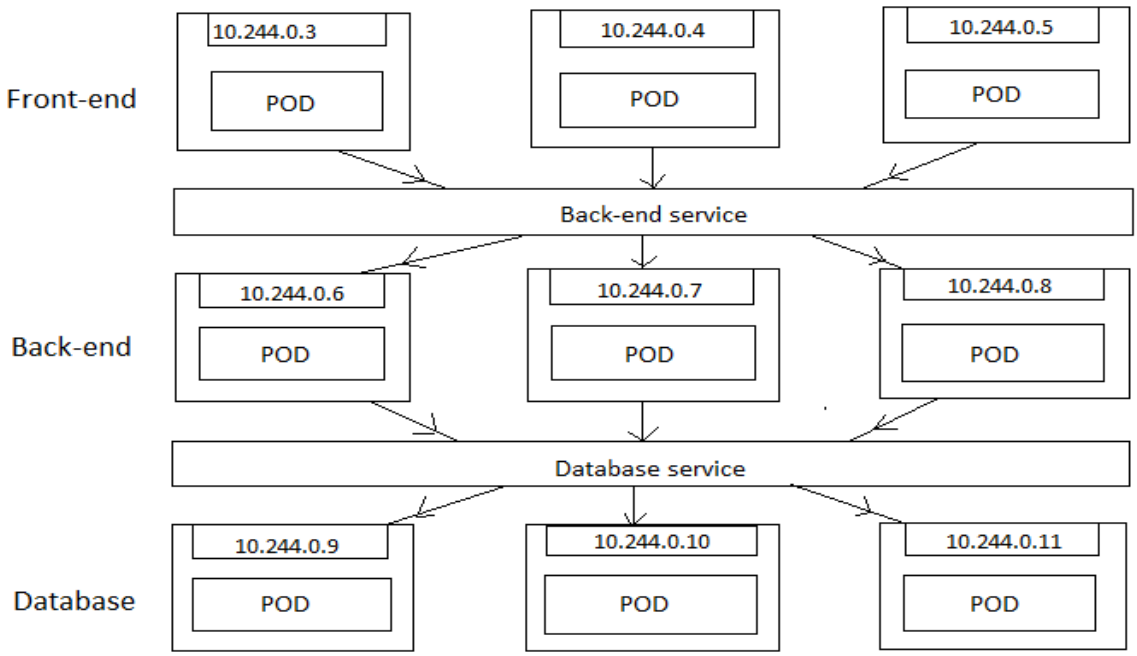


- In this case we have the web application on pods on separate nodes in the cluster.
- When we create a service without us having to do any additional configuration Kubernetes automatically creates a service that span across all the nodes in the cluster and maps the target port to the same node port on all the nodes in cluster. This way you can access your application using the IP of any node in the cluster and using the port number (NodePort No. 30008). Using the IP address of any of these nodes and trying to curl to the same port and the same port is made available on all the NodePorts of the cluster.
- In any case whether it be a single pod on a single node, multiple pods on a single node or multiple pods on multiple nodes the service is created exactly the same without having to do any additional steps during service creation.
- When pods are removed or added the service is automatically updated making its highly flexible and adaptive once created.

2) ClusterIP: -

- When pods are removed or added the service is automatically updated making its highly flexible and adaptive once created.
- The web front-end server needs to communicate to the back-end server and back-end server needs to communicates or connect to databases as well as the database (MySQL) services etc.

  The pods all have IP address assigned to them. But these Ip address as we know are not static. These pods can go down any time and new pod are created all the time and you can not relay on these IP addresses for internal communication between them.
- To establish connectivity between these services or tier of my application.
- A Kubernetes service can help us to group these pods together and provide a single interface to access the pods in a group.

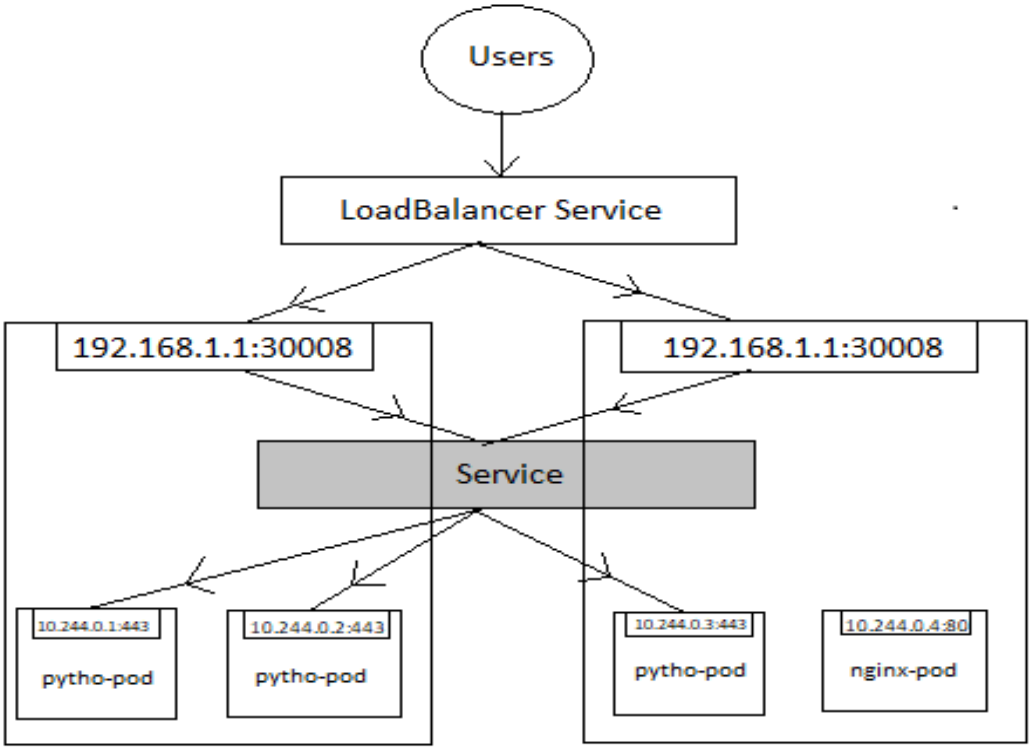


- This enables us to easily and effectively deploy a microservices based application on kubernetes cluster.
- Each layer can now scale or move as required without impacting communication between the various services. Each service gets an IP name assigned to it inside the cluster and that's the name that should be used by other pods to access the services. This type of service is known as ClusterIP.

service-definition.yml (Cluster IP)

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  type: ClusterIP
  ports:
  - targetPort: 80
    port: 80
    nodeport: 30008
    protocol: TCP
  selector:
    app: myapp
    type: front-end
```

3) LoadBalancer:



- If you create a LoadBalancer it creates a NodePort which then creates a ClusterIP.
- A LoadBalancer service is the standard way to expose a service to the internet. This will give you a single IP address that will forward all traffic to your service.
- There is no filtering, no routing, etc. This means you can send almost any kind of traffic to it, like HTTP, TCP, UDP, etc.
- LoadBalancer service is use if we would like to have a single IP which distributes requests using some method like round robin to all our external nodes IPs. It is built on top of a NodePort service

service-definition.yml

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  type: LoadBalancer
  ports:
  - targetPort: 80
    port: 80
    nodeport: 30008
    protocol: TCP
  selector:
    app: myapp
    type: front-end
```