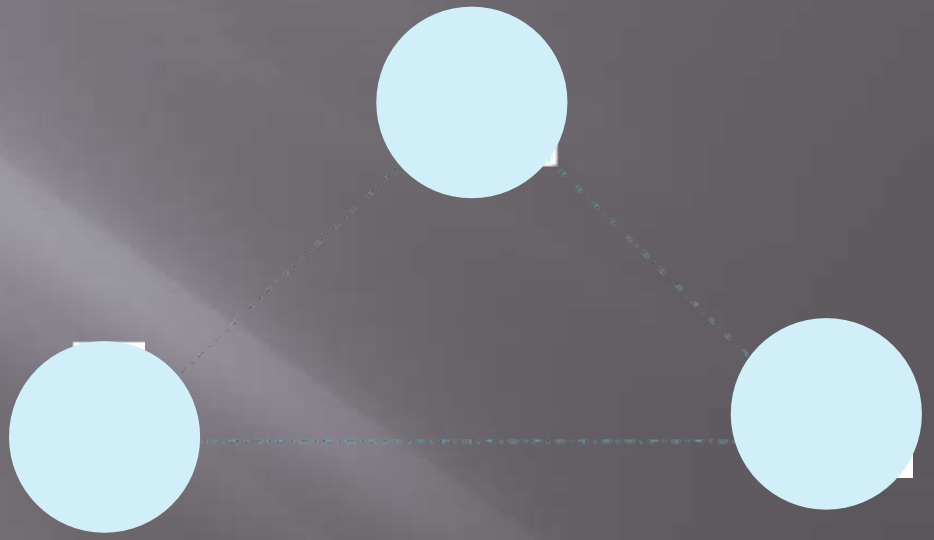
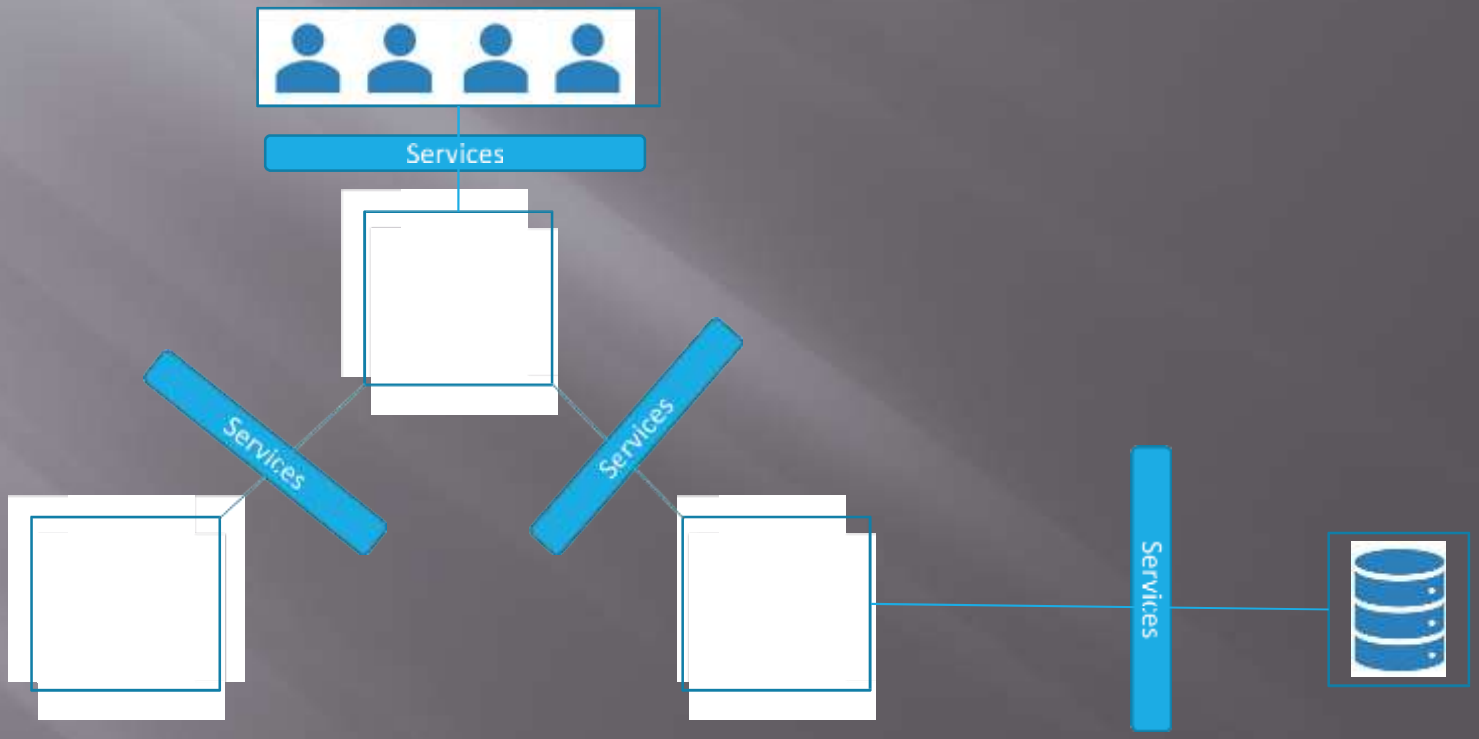
Services

Ravindra Kudache

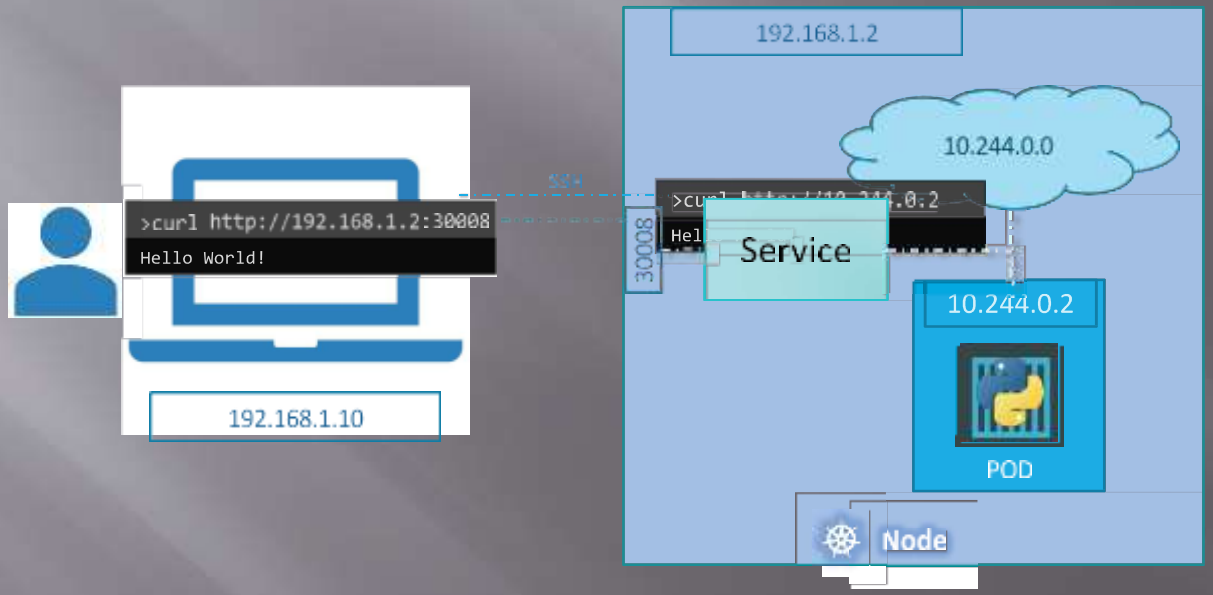
Services



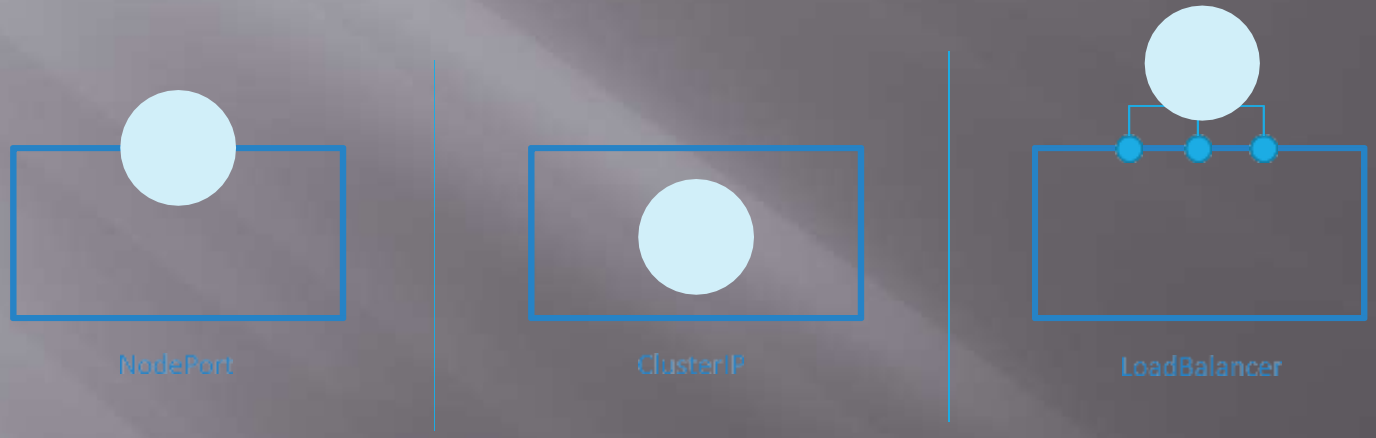
Services



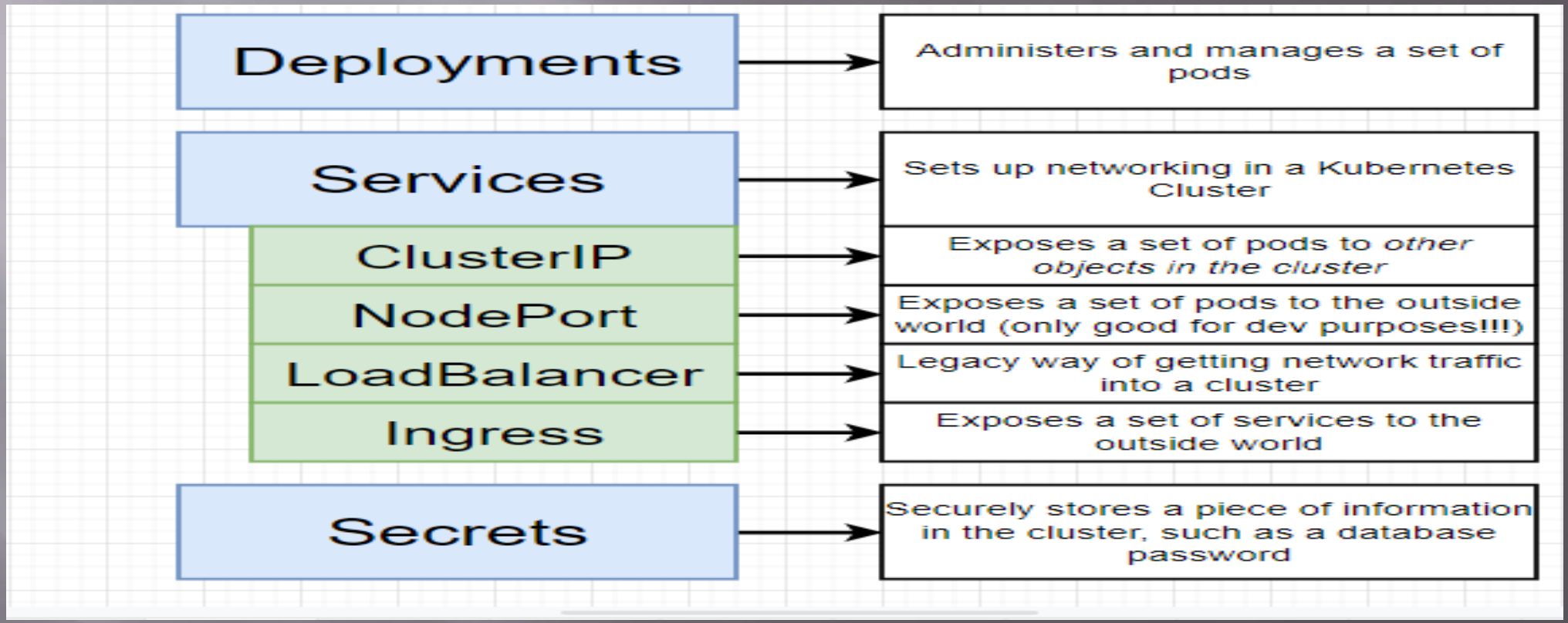
Service


>curl http://192.168.1.2:30008
Hello World!
192.168.1.10
192.168.1.2
10.244.0.0
30008
Service
10.244.0.2
POD
Node

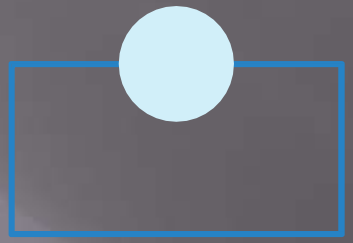
Services Types



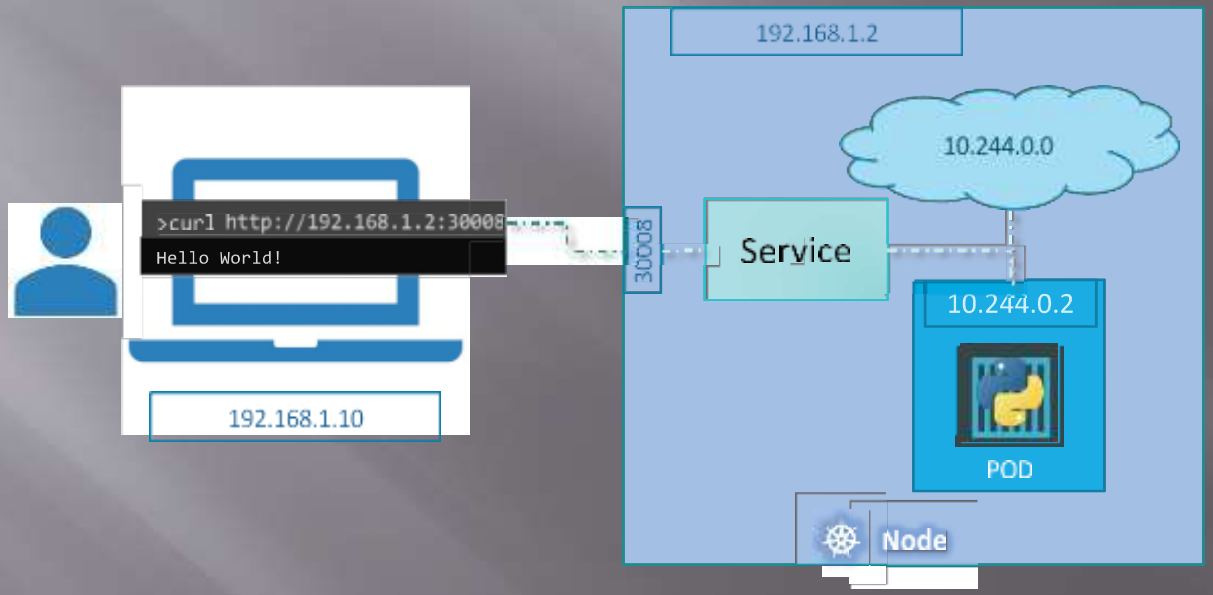
Services


Deployments
Administers and manages a set of pods
Services
Sets up networking in a Kubernetes Cluster
ClusterIP
Exposes a set of pods to *other objects in the cluster*
NodePort
Exposes a set of pods to the outside world (only good for dev purposes!!!)
LoadBalancer
Legacy way of getting network traffic into a cluster
Ingress
Exposes a set of services to the outside world
Secrets
Securely stores a piece of information in the cluster, such as a database password

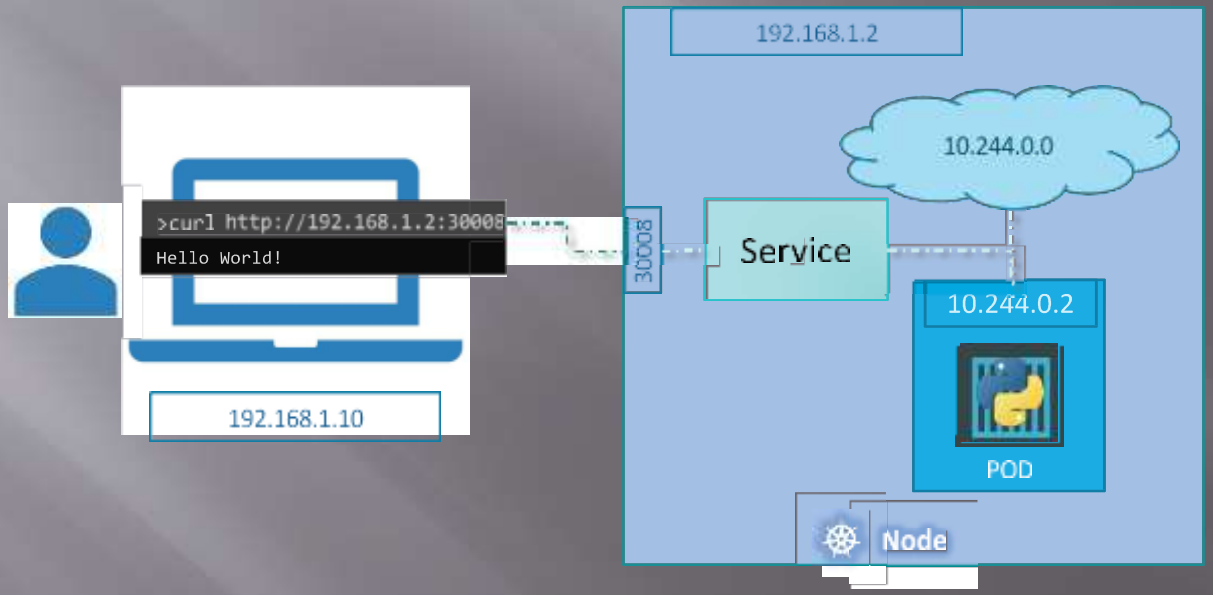
NodePort



Service - NodePort



Service - NodePort



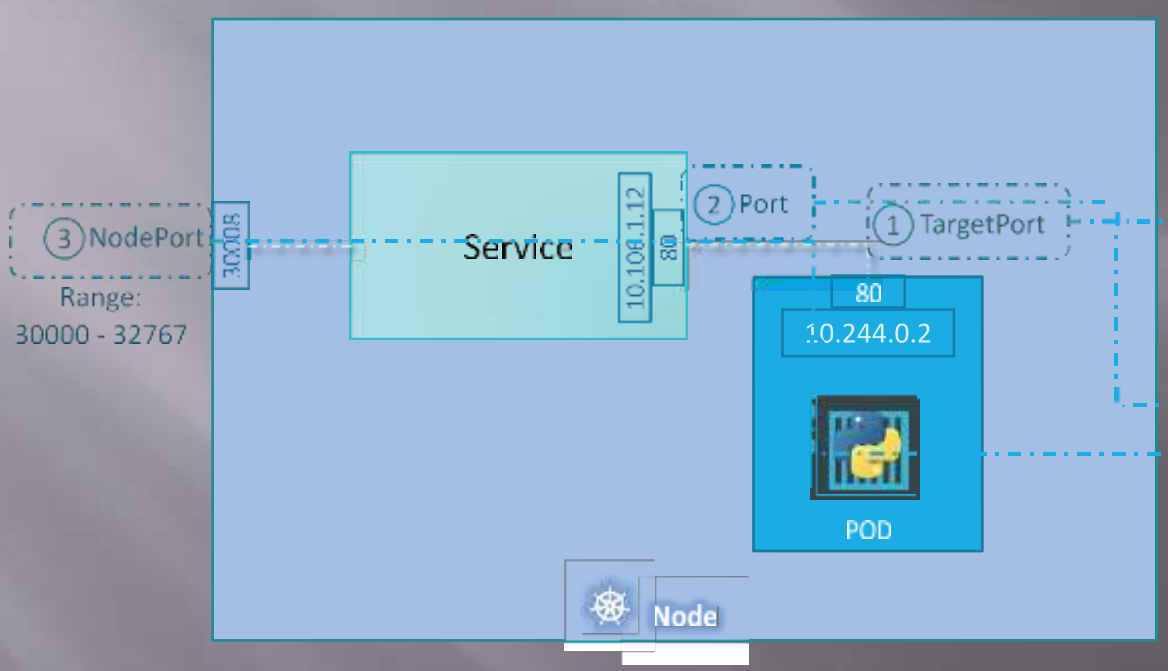
Service - NodePort

- Expose a container to the outside world (Only good for dev purposes!!)
- NodePort is exactly what it sounds like - makes it possible to access the app within the cluster using the IP of the Node (on which the Pod has been scheduled) and a random port assigned by Kubernetes e.g. for a HTTP endpoint, you would use http://<node_ip>:<port>
- The particular port is either specified or selected randomly from a non-privileged TCP port range between 30,000 and 32,767.

Service - NodePort

- Limitation:-
- NodePort services can't be called from outside of the AKS cluster since the cluster nodes or VM instances are provisioned with VNet private IP addresses and aren't directly exposed to the internet. In this respect NodePort services have limited functionality when used within AKS. Having said that a NodePort service is useful as a building block for testing services since you do not incur the extra expense and/or configuration when using for example the LoadBalancer service type.

Service - NodePort

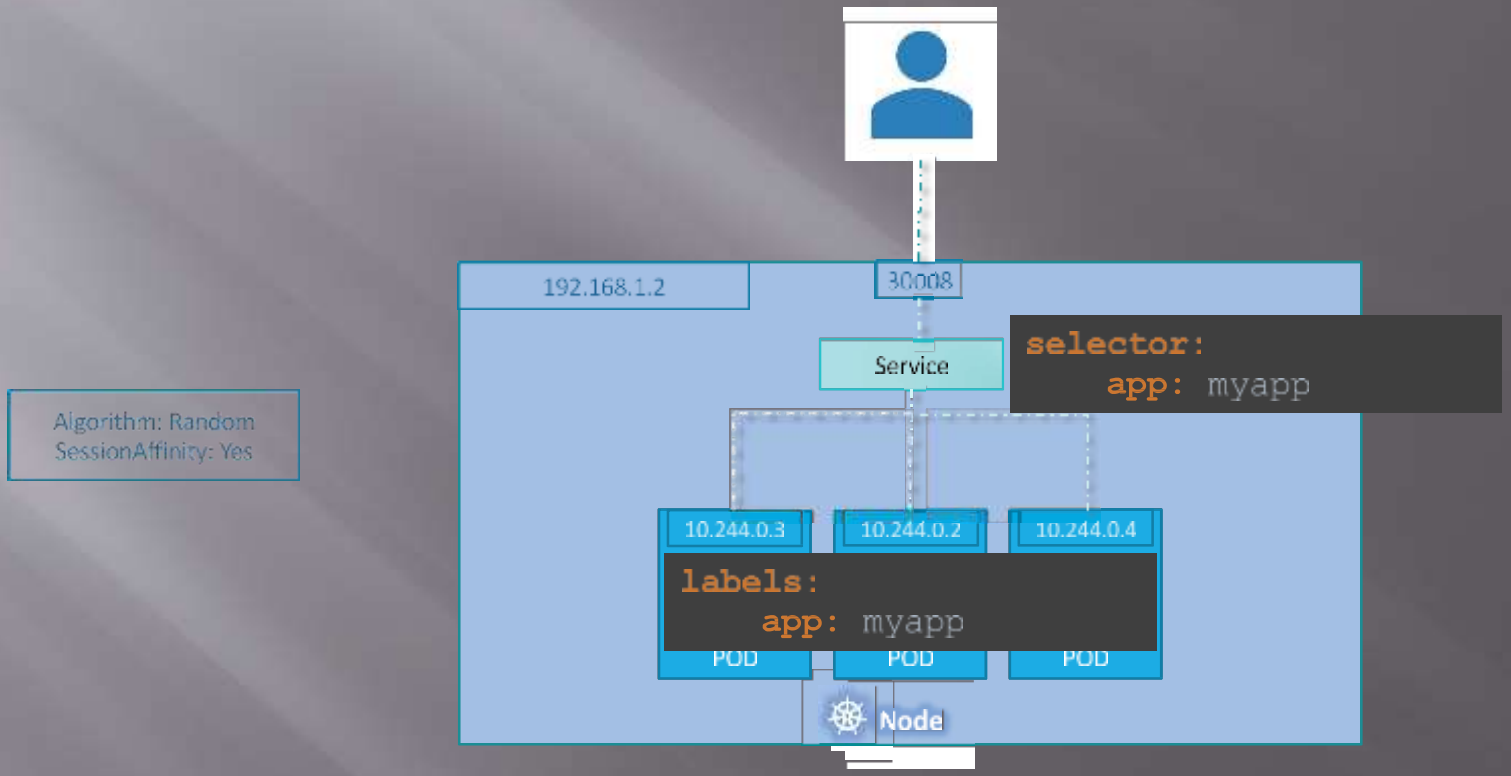


service-definition.yml

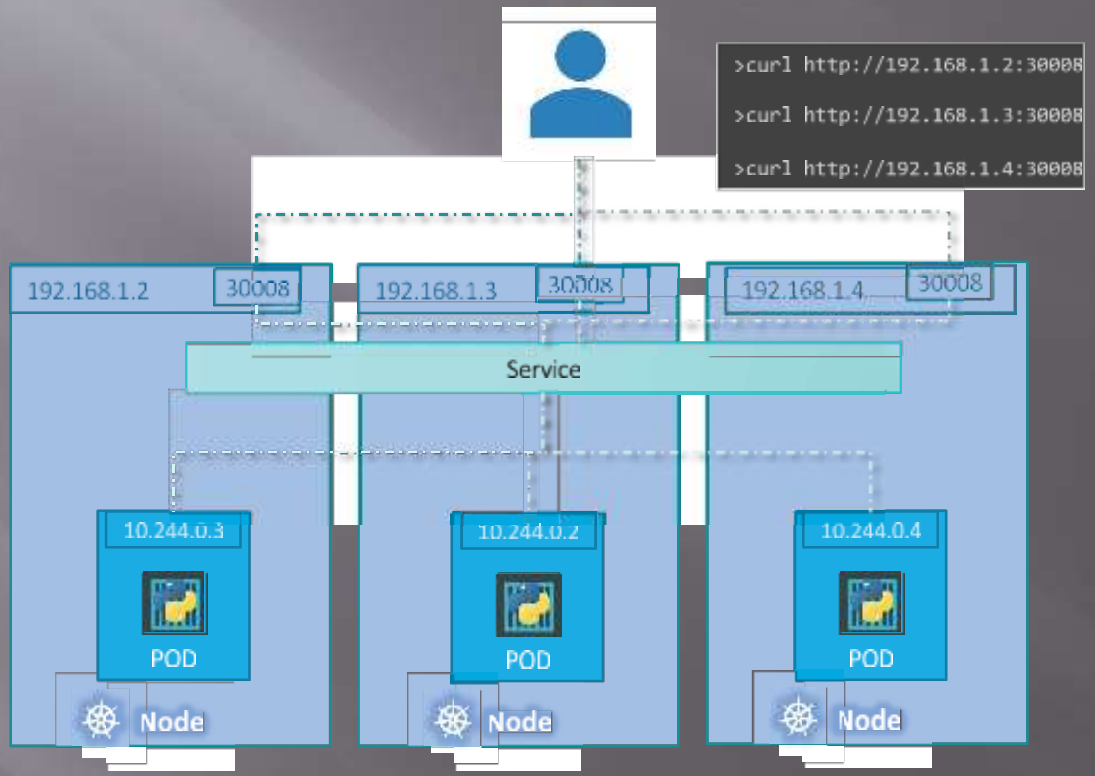
```yaml
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

spec:
    type: NodePort
    ports:
      - targetPort: 80
        port: 80
        nodePort: 30008
```

Service - NodePort



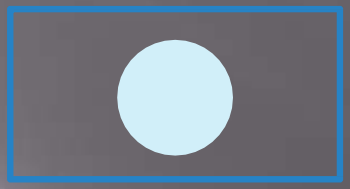
Service - NodePort



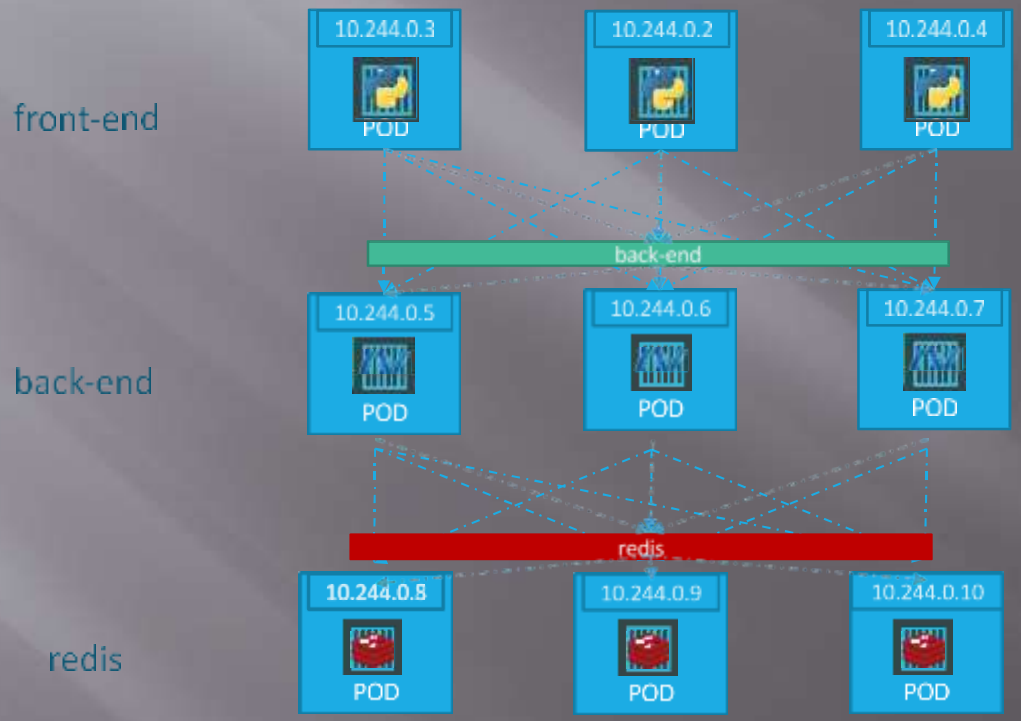
Demo

Service - NodePort

ClusterIP



ClusterIP


front-end
back-end
redis
10.244.0.3
10.244.0.2
10.244.0.4
POD
POD
POD
back-end
10.244.0.5
10.244.0.6
10.244.0.7
POD
POD
POD
redis
10.244.0.8
10.244.0.9
10.244.0.10
POD
POD
POD

ClusterIP

- Creating a Kubernetes service of type ClusterIP, results in a service being provisioned with an internally assigned IP address. ClusterIP services are the default service type. When you create a service without specifying its type, then Kubernetes will default it to ClusterIP.

- A ClusterIP service will be provisioned with an internal cluster VIP that can be called upon by any pods within the cluster and also from the nodes themselves. Outside of the cluster this IP is not callable. You'll find that all of the services that are deployed internally within the cluster inside the kube-system namespace are created this way.

service-definition.yml

```yaml
apiVersion: v1
kind: Service
metadata:
    name: back-end

spec:
    type: ClusterIP
    ports:
     - targetPort: 80
       port: 80

    selector:
```

pod-definition.yml

```
> kubectl create -f service-definition.yml
service "back-end" created

> kubectl get services
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP    16d
back-end      ClusterIP   10.106.127.123   <none>        80/TCP     2m
```

```yaml
      app: myapp
      type: back-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

Service


8080
Voting-app-Pod
6379
Redis
Service
6379
redis-pod
def get_redis():
if not hasattr(g, 'redis'):
g.redis = Redis(host="redis", db=0, password=redis_password, socket_timeout=5)
return g.redis
Not specified in source code.
Default 6379 assumed
apiVersion: v1
kind: Service
metadata:
name: redis
spec:
ports:
- port: 6379
targetPort: 6379
selector:
name: redis-pod
app: demo-voting-app

Service


8080
Voting-app-Pod
7000
Redis
Service
6379
redis-pod
def get_redis():
if not hasattr(g, 'redis'):
g.redis = Redis(host="some-redis-service", port=7000, db=0, password=redis_password, socket_timeout=5)
return g.redis
apiVersion: v1
kind: Service
metadata:
name: some-redis-service
spec:
ports:
- port: 7000
targetPort: 6379
selector:
name: redis-pod
app: demo-voting-app

References

https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/