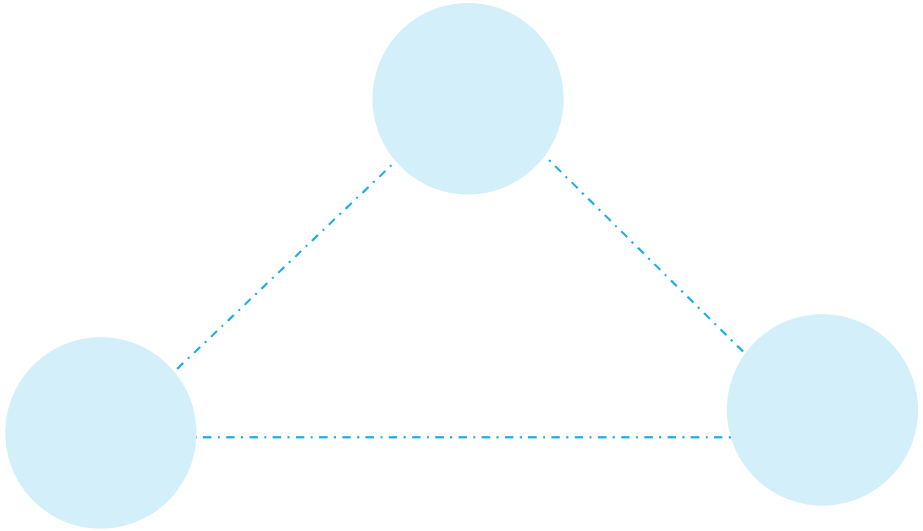
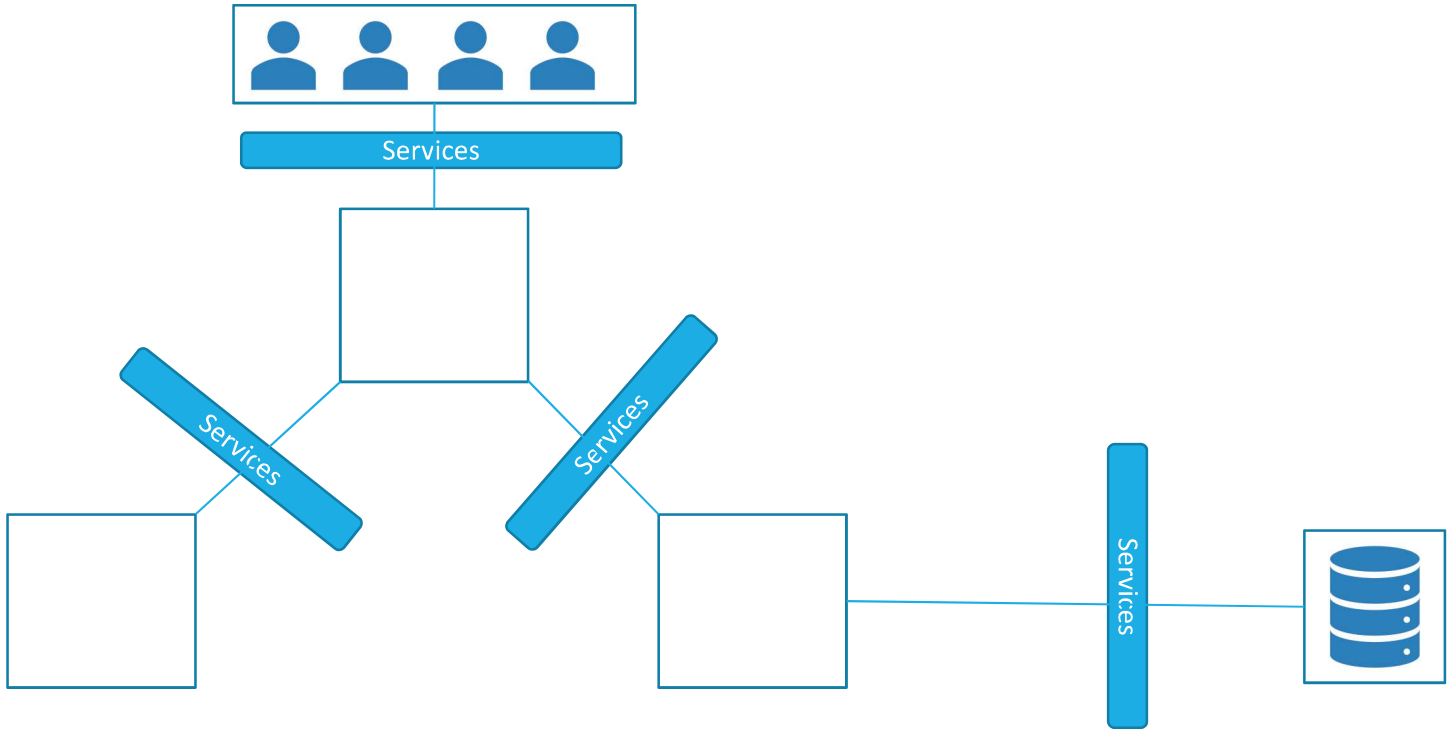
Services



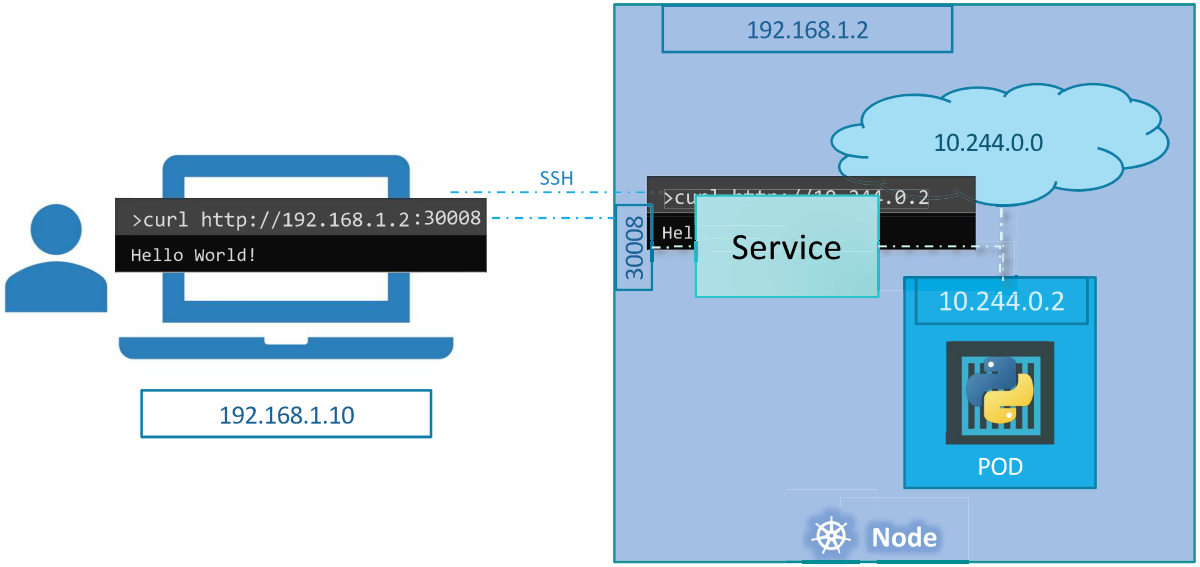
Services



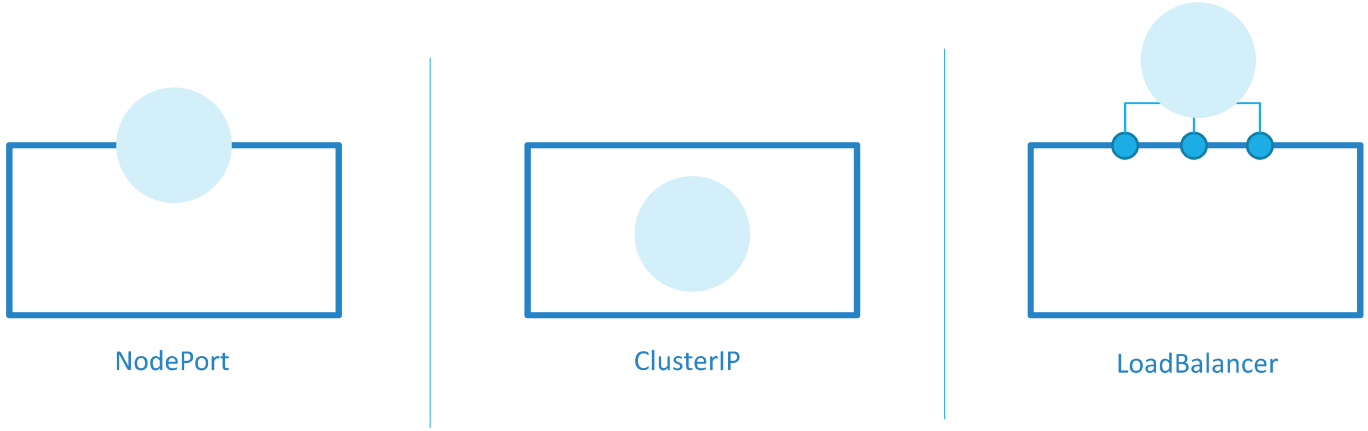
I cannot just reach the Pod from the outside, since it is on another network. I'd have to ssh into the node first.

I need a service that listens on a Port on the node and forwards the requests to the Pod. This is a NodePort Service.

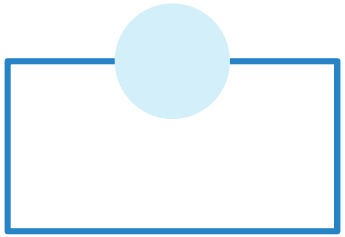
Service



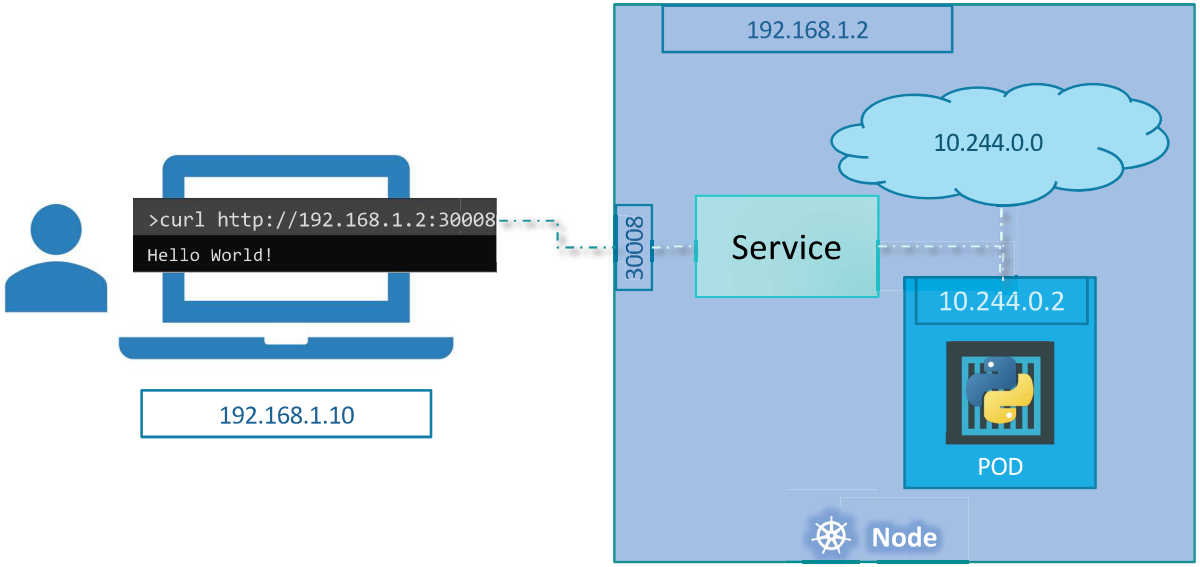
Services Types



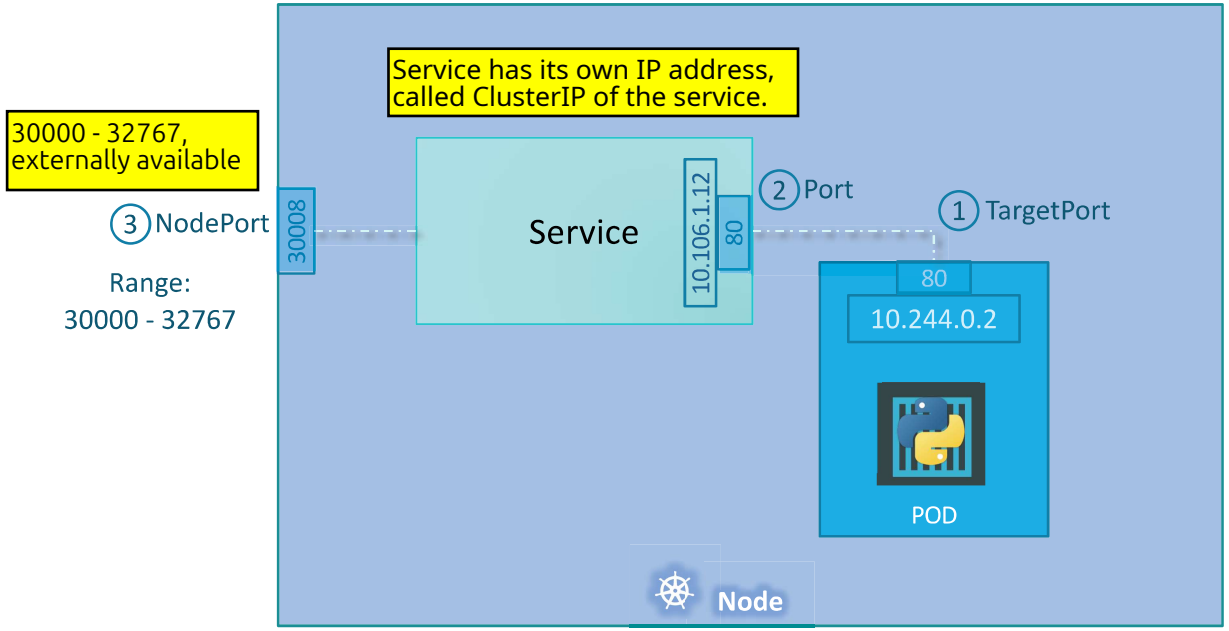
NodePort



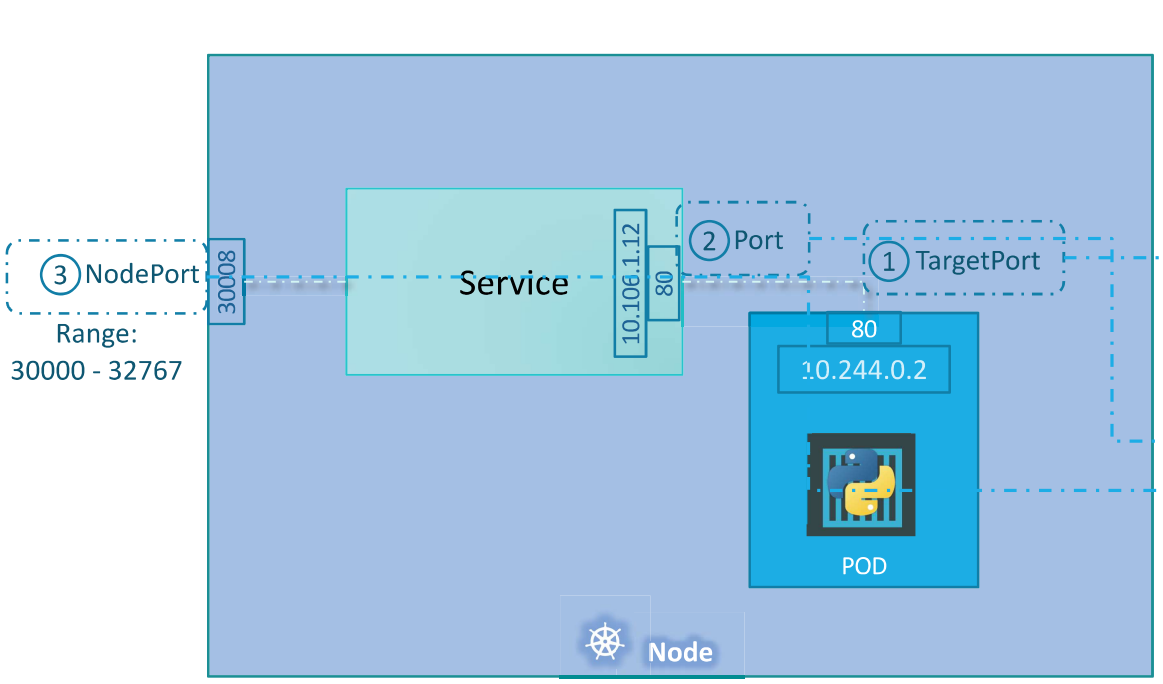
Service - NodePort



Service - NodePort



Service - NodePort



service-definition.yml

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

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

Selector is used to tell service which cPods it should forward requests to. This links the Service to the Pods.

Service - NodePort

service-definition.yml

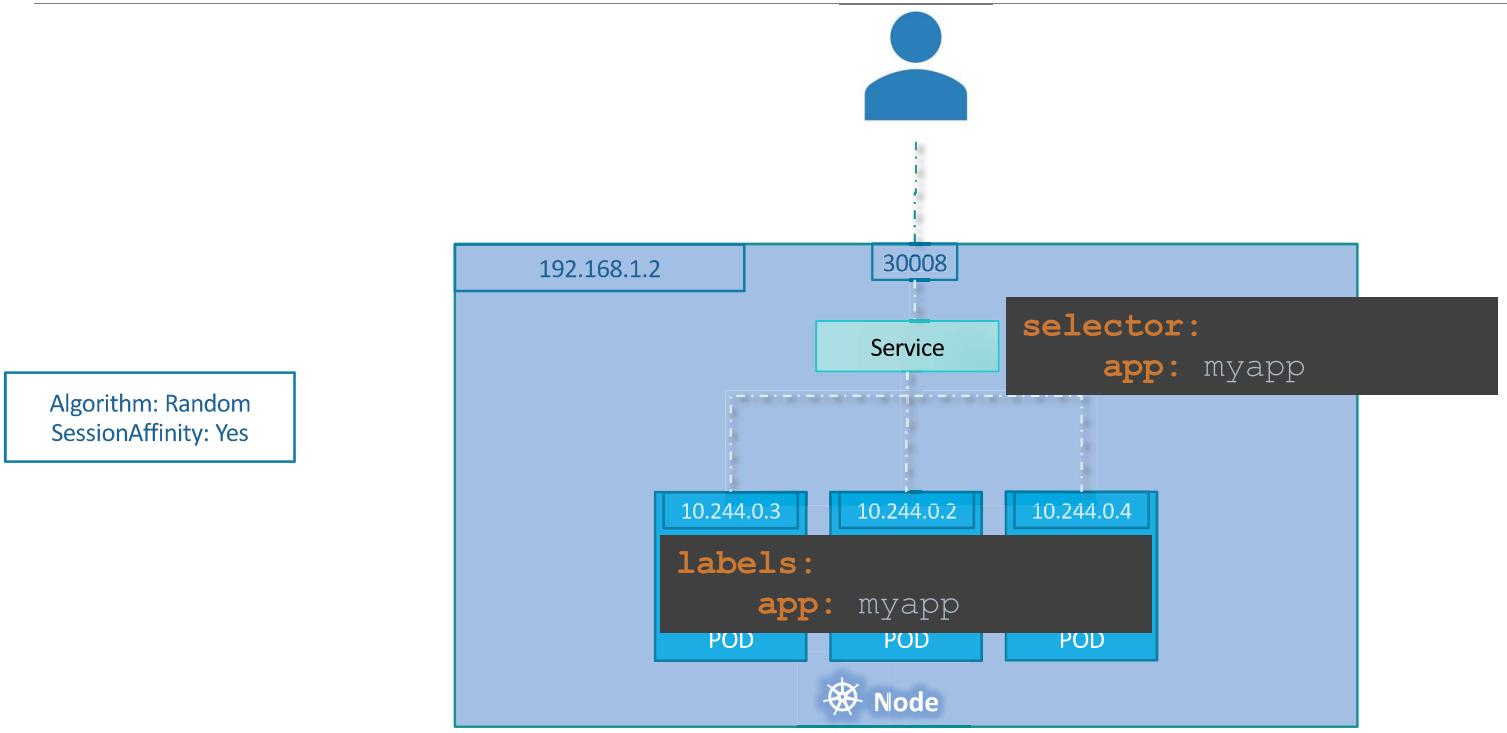
```
apiVersion: v1
kind: Service
metadata:
    name: myapp-service

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

the only required field is port

pod-definition.yml

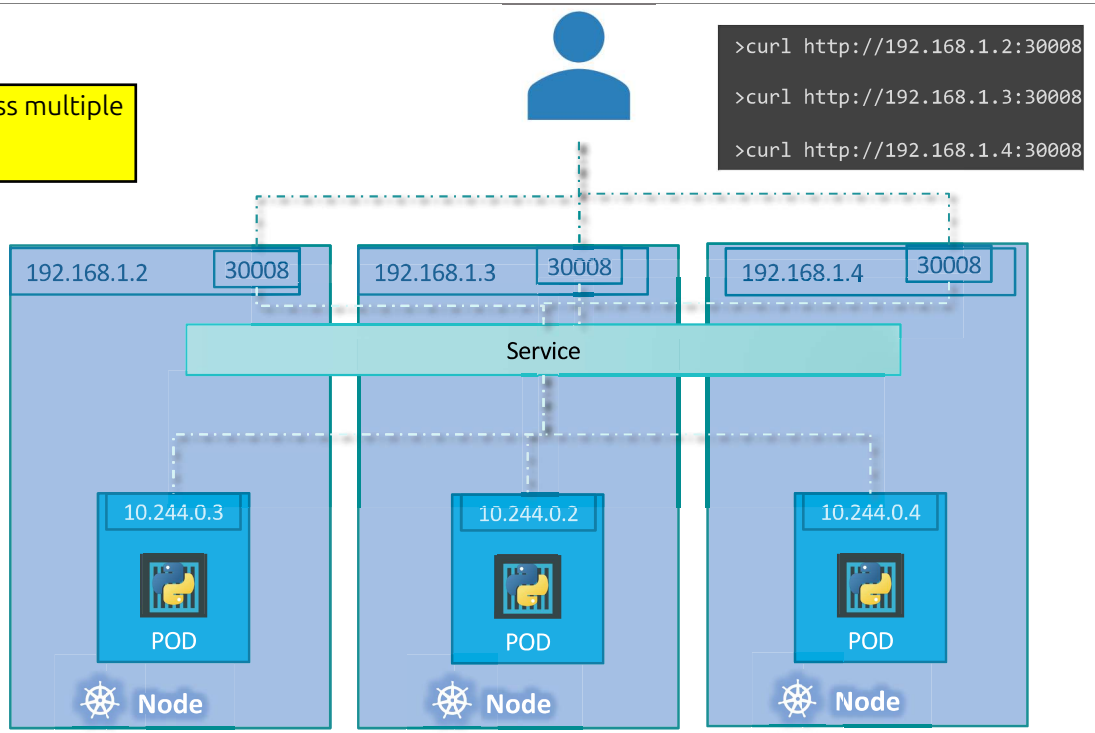
```
> kubectl create -f service-definition.yml
service "myapp-service" created

> kubectl get services
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP        16d
myapp-service   NodePort    10.106.127.123   <none>        80:30008/TCP   5m

      app: myapp

> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
```

Service - NodePort



Service - NodePort

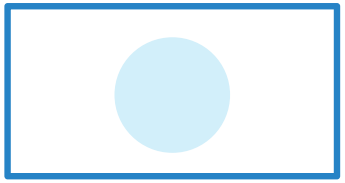
A service can automatically span across multiple Nodes. Awesome!



Demo

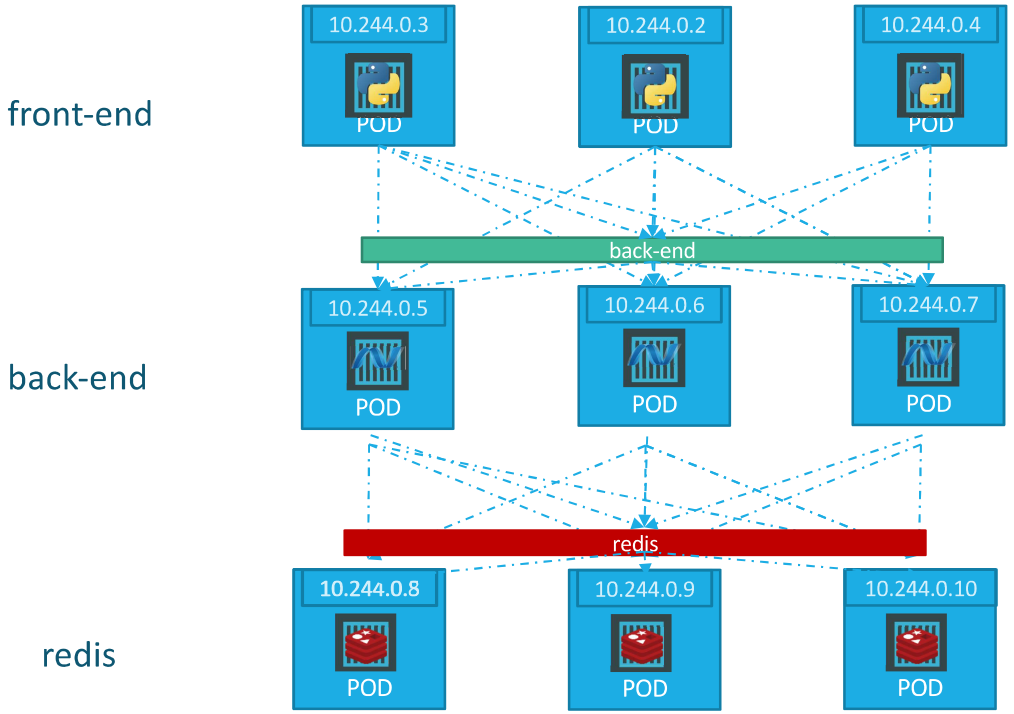
Service - NodePort

ClusterIP



Kubernetes Service ClusterIP groups the back-end Pods together and the backend application workload can just be reached by using the service name 'back-end'

ClusterIP



service-definition.yml

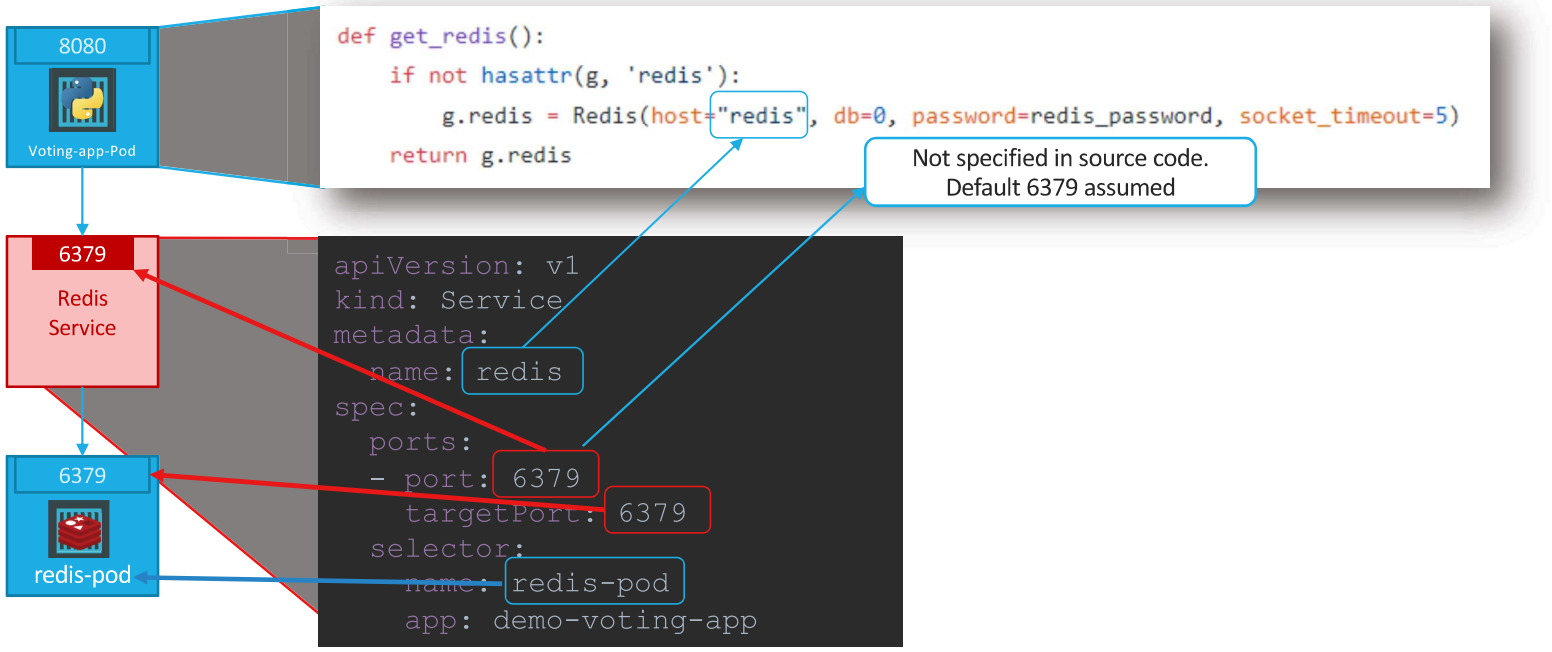
```yaml
apiVersion: v1
kind: Service
metadata:
    name: back-end

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

    selector:
```

pod-definition.yml

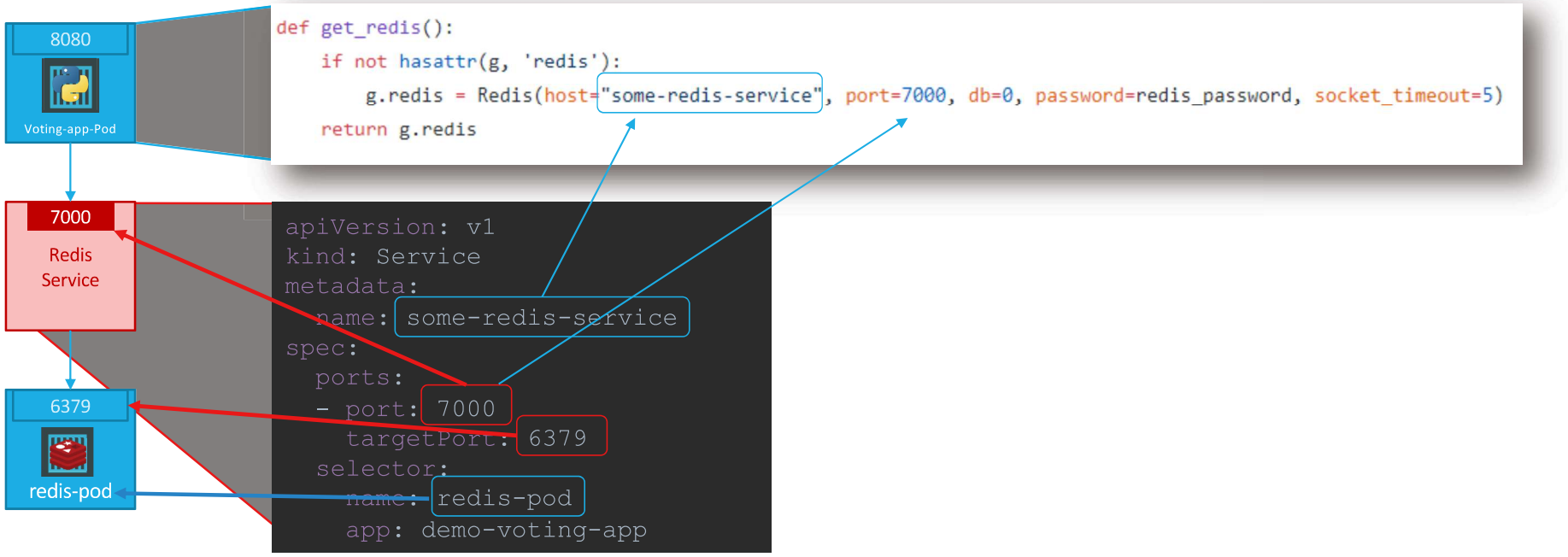
```
> kubectl create -f service-definition.yml
service “back-end" created

> 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

Demo

Service - NodePort

References

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