USING KUBERNETES PORT, TARGETPORT, AND NODEPORT

(This article is part of our Kubernetes Guide. Use the right-hand menu to navigate.)

Port configurations for Kubernetes Services

In Kubernetes there are several different port configurations for Kubernetes services:

- **Port** exposes the Kubernetes service on the specified port within the cluster. Other pods within the cluster can communicate with this server on the specified port.
- **TargetPort** is the port on which the service will send requests to, that your pod will be listening on. Your application in the container will need to be listening on this port also.
- **NodePort** exposes a service externally to the cluster by means of the target nodes IP address and the NodePort. NodePort is the default setting if the port field is not specified.

Let's look at how to use these ports in your Kubernetes manifest.

Interested in Enterprise DevOps? Learn more about DevOps Solutions and Tools with BMC. ›

Using Port, TargetPort, and NodePort

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: hello-world
spec:
  type: NodePort
  selector:
    app: hello-world
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
      nodePort: 30036
```

From the above examples the hello-world service will be exposed internally to cluster applications on port 8080 and externally to the cluster on the node IP address on 30036. It will also forward requests to pods with the label “app: hello-world” on port 80.

The configuration of the above settings can be verified with the command:

```
$ kubectl describe service hello-world
```

Create a pod running nginx to which the service will forward requests to:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: hello-world
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 80
```

To test and demonstrate the above configuration, we can create a pod running an ubuntu container to execute some curl commands to verify connectivity.

```
$ kubectl run -i --tty ubuntu --image=ubuntu --restart=Never -- sh
```

From this pod run the following commands:

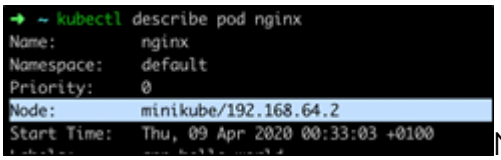
Curl the service on the 'port' defined in the Kubernetes manifest for the service.

```
$ curl hello-world:8080
```

This proves that curling the Kubernetes service on port 80 forwards the request to our nginx pod listening on port 80.

To test the **NodePort** on your machine (not in the ubuntu pod) you will need to find the IP address of the node that your pod is running on.

```
$ kubectl describe pod nginx
```



Now, you can curl the **Node IP Address** and the **NodePort** and should reach the nginx container running behind the Kubernetes service.

Additional resources

For more on Kubernetes, explore these resources:

- Kubernetes Guide, with 20+ articles and tutorials
- BMC DevOps Blog
- Bring Kubernetes to the Serverless Party
- How eBay is Reinventing Their IT with Kubernetes & Replatforming Program