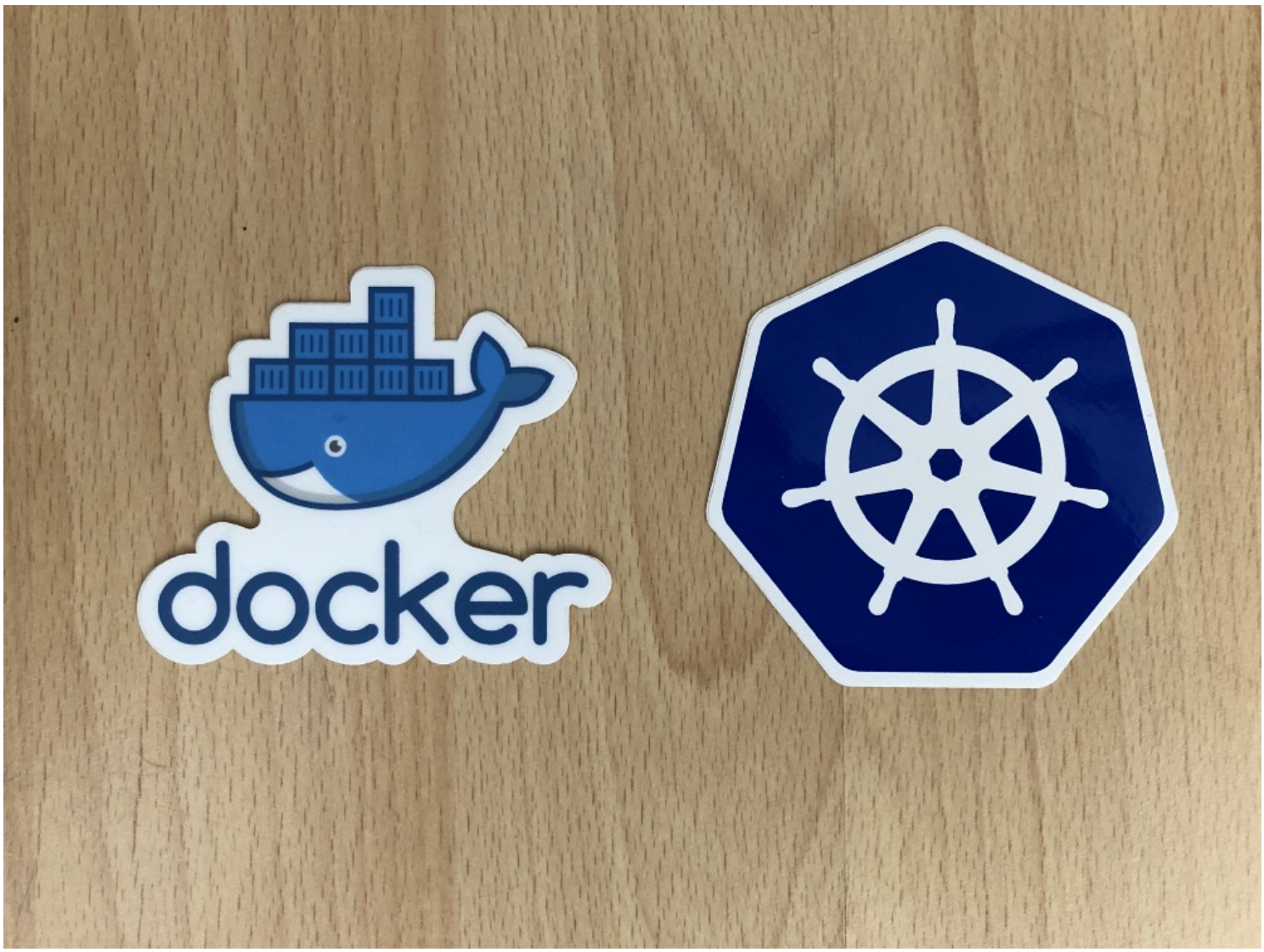
Liz Rice

Follow 1.8K Followers About



Accessing an application on Kubernetes in

Liz Rice Feb 12, 2018 · 3 min read

There are some good tutorials to get you started with Kubernetes on Docker for Mac / Windows, like this one from Romin Irani. Once you've got your application running in Kubernetes, there are a couple of ways to access it from your desktop.

Port forwarding

Port forwarding directs traffic to a port in a particular pod:

```
$ kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
aqua-csp-766689745b-4ln7q   1/1       Running   0          6m

$ kubectl port-forward aqua-csp-766689745b-4ln7q 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
```

At this point I can browse to 127.0.0.1:8080 and access the Aqua Security web service I've got running on my Kubernetes deployment.

Port forwarding lets you access a **specific pod**'s container port directly. If you've got just one pod (perhaps because it's a deployment with a single replica), this is a straightforward way to access it. It's also the approach to use if you have reason to access a particular pod.

You will need to leave this command running though, so you'll want to background it (or just start another terminal window).

Exposing a service

Alternatively you might want to expose a service associated with your deployment. Oftentimes your project's YAML files will already have a service definition, but if the service doesn't already exist you can easily create one with `kubectl expose deployment`:

```
$ kubectl expose deployment aqua-csp --type=NodePort --name=aqua-csp
service "aqua-csp" exposed
```

```
aqua-csp     NodePort    10.102.47.85   <none>        8080:32407/TCP
12s
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP
18d
```

I can now browse to localhost:32407 and find my web UI.

If you're making changes to application code and redeploying your pods, with port forwarding you'd need to re-run the port forwarding command each time you recreate the pod. A service, on the other hand, can stay in place while the pods it routes traffic to get recreated.

If you're using YAML files from your production deployment, be aware that a service of type LoadBalancer isn't going to work on your local machine. You can create the service, but the External IP address will stay in Pending state indefinitely. This is because LoadBalancer type relies on an external provider (like your cloud service) setting up a load balancer for sending traffic to the service. You'll need to set the type to NodePort.

Got no port?

Suppose you see an error like this:

```
$ kubectl expose deployment webserver --type=NodePort
error: couldn't find port via --port flag or introspection
```

Checking out the deployment shows us that there is no container port specified:

```
$ kubectl describe deployment webserver
Name:                   webserver
...
Pod Template:
  Labels:  k8s-app=webserver
  Containers:
   webserver:
    Image:        nginx:alpine
    Port:         <none>
...
```

```
$ kubectl expose deployment webserver --type=NodePort --port=80
service "webserver" exposed

$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP
19d
webserver    NodePort    10.111.216.219   <none>        80:30488/TCP
7s
```

. . .

Updated to reflect philoserf's suggestion that it would be a good idea to explain when and why you might take either approach. Also added info about specifying the container port.

Kubernetes Docker