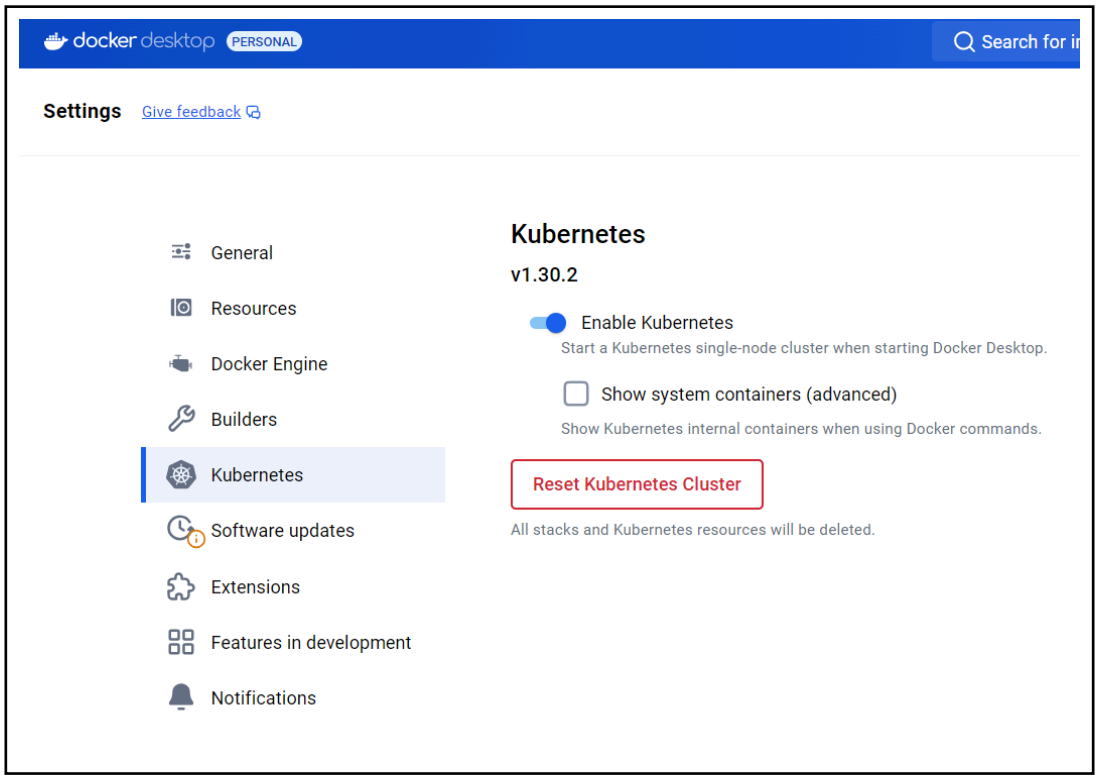
Hello Kubernetes

Goals

1. Start the built-in version of Kubernetes in Docker Desktop
2. Build a custom Docker image for a web app
3. Create a Kubernetes deployment for the web app
4. Create a Kubernetes service to access the web app

Kubernetes in Docker Desktop

Docker Desktop now comes with a built-in version of Kubernetes. To start it, make sure that you have it selected in the `Settings` control:



You can see if Kubernetes is running in the bottom left of Docker Desktop. If you have trouble, you may have to click the `Reset Kubernetes Cluster` button.

Building a Custom Docker Image

At this point, we've built custom Docker images multiple times so this should be familiar to you. Our `Dockerfile` as well as other supporting files are in the `exercises/hello-k8s` directory of the class git repo.

```
$ cd exercises/hello-k8s
exercises/hello-k8s$ ls
Dockerfile  elf11.zip  index.html
exercises/hello-k8s$ docker build -t elf:v1 . ❶
<snip>
=> exporting to image                                                                    0.0s
=> => exporting layers                                                                   0.0s
=> => writing image sha256:01eee4b43dd28560b1a0f7dfa57957baa07772559065454f5f42a957d95b7031 0.0s
=> => naming to docker.io/library/elf                                                    0.0s
```

Note that we're also giving this image a version tag (v1). Kubernetes in Docker Desktop requires this to load a local image.

Creating a Kubernetes Deployment

Now we'll use kubectl to create a deployment in Kubernetes, check to make sure it's running, and see if it started any pods:

```
$ kubectl create deployment hello-k8s --image=elf:v1
deployment.apps/hello-k8s created
$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
hello-k8s   1/1     1            1           2m42s
$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
hello-k8s-b9f677dd8-mvtts    1/1     Running   0          7s
```

Notice that our deployment, named `hello-k8s`, caused a pod to start running.

Creating a Kubernetes Service

We have a deployment running and that deployment has started a pod to run our Docker image. If we want to be able to connect to the pod, we will need to set up a `service`. Let's use the `kubectl expose` command to make that service:

```
$ kubectl expose deployment hello-k8s --type=LoadBalancer --port=8000
service/hello-k8s exposed
$ kubectl get services
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
hello-k8s    LoadBalancer   10.100.129.66   localhost     8000:30246/TCP   36s
kubernetes   ClusterIP      10.96.0.1       <none>        443/TCP          100m
```

Deliverables

Now that you have your web app up and running, open a web browser and go to http://localhost:8000 . All you have to submit in the textbox for this assignment is the *name* of the the game you're playing!

Wrapping Up

Now by running these `kubectl` commands you've created a few objects with default values. You can actually view *all* the options for the things that were created with the `get -o yaml` command which literally means print out the YAML that defines that object. Here's some commands you can try:

```
$ kubectl get -o yaml deployment hello-k8s
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  creationTimestamp: "2024-10-21T18:49:27Z"
<snip>
$ kubectl get -o yaml service hello-k8s
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2024-10-21T20:27:23Z"
  labels:
    app: hello-k8s
  name: hello-k8s
<snip>
```

As you can see there are a lot of options for these objects!

To delete the `Deployment` and the `Service`, which will delete the one `Pod` created, use the following commands:

```
$ kubectl delete deployment hello-k8s
deployment.apps "hello-k8s" deleted
$ kubectl delete service hello-k8s
service "hello-k8s" deleted
```