Lab:
Using the `kubectl` CLI

Testing `kubectl` Cluster Access

- Open your codespace if it has closed
- Verify minikube is running with: `minikube status`
    - If it is not, start minikube: `minikube start`
- Test the access to your cluster by using `kubectl` commands:
    - `kubectl cluster-info`
        - This will return information about the cluster
    - `kubectl get services`
        - This should return a single service named Kubernetes
    - `kubectl get nodes`
        - This should return a single minikube node
    - `kubectl get pods`
        - This should not return any pods

`kubectl` CLI

- `kubectl` uses a config file located in the `.kube` folder of your home directory to know which cluster to connect to
- The previous commands worked because when you created the minikube cluster it automatically created the `kubectl config` file
  - Feel free to investigate that file, but be sure not to modify it
- If anything does happen to that file, or if you want to use the kubectl CLI on another system, you would need to create the `.kube/config` file

Deploying a Pod with `kubectl`

- From the terminal, make a new folder

  ```
  cd /workspaces/eventsapp/
  mkdir kubernetes-config
  ```

- In the `kubernetes-config` folder, create a new file named `mario-pod.yaml`
  - You can create the file with nano, vi, or the visual editor
  - Paste in the contents shown here

Just for fun, we are deploying a container image that is a Super Mario web game

```
apiVersion: v1
kind: Pod
metadata:
  name: supermario-pod
  labels:
    app: mario
spec:
  containers:
  - name:  supermario-demo
    image: pengbai/docker-supermario
    ports:
    - containerPort: 8080
```

Deploying a Pod with `kubectl` (continued)

- Run the following commands:

```
kubectl apply -f mario-pod.yaml
kubectl get pods
```

- You should see the following output:

```
$ kubectl apply -f mario-pod.yaml
pod/supermario-pod created
$ kubectl get pods
NAME              READY    STATUS     RESTARTS    AGE
supermario-pod    1/1      Running    0           4s
```

If the pod is not yet Ready, run `kubectl get pods` again

- Run the following command to get more details on the pod:

```
kubectl describe pod supermario-pod
```

Exposing a Pod

- Pods can be exposed to the internet with a load balancer service
  - We will discuss this more later
  - Execute the following single command to expose the pod on port 80:
    ```
    kubectl expose pod supermario-pod --type=LoadBalancer \
    --name=supermario-svc --port=80 --target-port=8080
    ```
  - Verify the load balancer service was created:
    ```
    kubectl get service
    ```
  - Notice the name of the service is **`supermario-svc`**

Testing the Application

- If Minikube was running on a local system, you could test the app locally
  - Since you are running it on a remote environment (codespaces), you must create a port forward to be able to test the app
- Open a new terminal (+ button) and run the following command:
  - `minikube tunnel & kubectl port-forward service/supermario-svc 8080:80`
  - This is a blocking command—you will not get the prompt back
  - Be sure to leave it running
- You should get a popup, click the **Open in browser** button
  - If you don't see the **Open in browser** button, click on the **Ports** tab, hover over the line for port 8080, and click the globe (open in browser)
- You should see the Super Mario game load
  - If you want to play: Press **S** to start the game, cursor keys to navigate, S to enter the level, and use S to jump

Investigating the Pod

- Switch back to the terminal that is not running the port forward
- View the pod logs
```
kubectl logs supermario-pod
```
- Perform a directory listing of the pod's `WORKDIR`
```
kubectl exec supermario-pod -- ls -l
```
- Open a bash session inside the pod
```
kubectl exec -i -t supermario-pod -- /bin/bash
```
- Inside the bash session, try to access a site on the internet
```
curl http://cheat.sh
```
- Inside the bash session, view running processes in the container
```
ps ax
```
- Exit the bash session
```
exit
```

How to Stop the Port Forward

- In the terminal window running the port forward, press **`CTRL+C`** to stop it
- Then run the following command to stop the tunnel:

```
pkill -f "minikube tunnel"
```

Clean Up

- Delete the pod

```
kubectl delete pod supermario-pod
kubectl get pods
```

- The pod should terminate (it may take a few seconds to delete)
- Delete the service

```
kubectl delete service supermario-svc

kubectl get svc
```

- The `supermario-svc` service should now be deleted
  - The Kubernetes service should still be running—that is required by Kubernetes

Syncing the Changes to Git

- Commit these changes to your Git repository
  - On the left side, click the **Source Control** button
  - Be sure ALL changes are staged by clicking in the + button
  - Type a commit message of: **`Added Events app start code`** and click the **Commit** button
  - Press the **Sync Changes** button and press **OK** to push the changes
- The code has now been saved to your **eventsapp** Git repository created earlier

Success

- **Congratulations**! You have successfully configured the `kubectl` CLI for your cluster
  - Deployed a pod to your cluster with the `kubectl` CLI
  - Investigated various `kubectl` commands to interact with your cluster
  - Executed commands within a container running in a pod
  - Deleted pods