Lab:
Creating a ClusterIP and Load Balancer

Verify Minikube Is Running

- From a codespaces terminal, check if Minikube is still running with:
  `minikube status`
- If it is stopped, start it again with:
  `minikube start`

Verify the API and Web Deployments Are Running

- In your codespaces terminal, verify your deployments are running:
  `kubectl get deploy`

```
$ kubectl get deploy
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
events-api       1/1     1            1           4m42s
events-web       1/1     1            1           4m36s
```

- If your deployments are not running, change into your `kubernetes-config` folder and apply the two yaml files:

```
cd /workspaces/eventsapp/kubernetes-config
kubectl apply -f api-deployment.yaml
kubectl apply -f web-deployment.yaml
```

Creating the `api-service.yaml`

- In your codespaces terminal, change into your `kubernetes-config` folder:
  `cd /workspaces/eventsapp/kubernetes-config`
- In the `kubernetes-config` folder, create a new file named `api-service.yaml`
  - Copy/paste the contents shown here:
- Which pods will be added to this service?

```yaml
apiVersion: v1
kind: Service
metadata:
 labels:
   app: events-api-svc
 name: events-api-svc
spec:
 ports:
 - port: 8082
   protocol: TCP
   targetPort: 8082
 selector:
   app: events-api
   ver: v1.0
 type: ClusterIP
```

Deploying the `api-service.yaml`

- Deploy the `events-api` service:

  `kubectl apply -f api-service.yaml`

- Verify it was deployed:
  `kubectl get service`

```
$ kubectl get service
NAME             TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
events-api-svc   ClusterIP   10.8.10.141   <none>        8082/TCP   4m1s
kubernetes       ClusterIP   10.8.0.1      <none>        443/TCP    4h49m
```

- Notice the service name is `events-api-svc`
  - That name will be resolved to the `cluster-ip` by the kube DNS

Modify the `web-deployment.yaml`

- Remember, the `events-website` service uses an environment variable (called `SERVER`) to know where to connect for the `events-api` service
  - Environment variables can be set on a Kubernetes deployment
  - In this case, the hostname can be set to the name of the `events-api-svc`
- Edit your `web-deployment.yaml` file
  - At the end of the file, add the three highlighted lines shown here:
  - Be sure to note the indentation
- Then reapply the file:
  `kubectl apply -f web-deployment.yaml`

Add these three lines

```
containers:
- image: IMAGE-URL
  name: events-web
  ports:
  - containerPort: 8080
  env:
  - name: SERVER
    value: "http://events-api-svc:8082"
```

Creating the `web-service.yaml`

- In the `kubernetes-config` folder, create a new file named `web-service.yaml`
  - Copy/paste the contents shown here:

```
apiVersion: v1
kind: Service
metadata:
 labels:
   app: events-web-svc
 name: events-web-svc
spec:
 ports:
 - port: 80
   protocol: TCP
   targetPort: 8080
 selector:
   app: events-web
   ver: v1.0
 type: LoadBalancer
```

Deploying the `web-service.yaml`

- Deploy the `events-web` service:
  `kubectl apply -f web-service.yaml`
- Verify it was deployed:
  `kubectl get service`

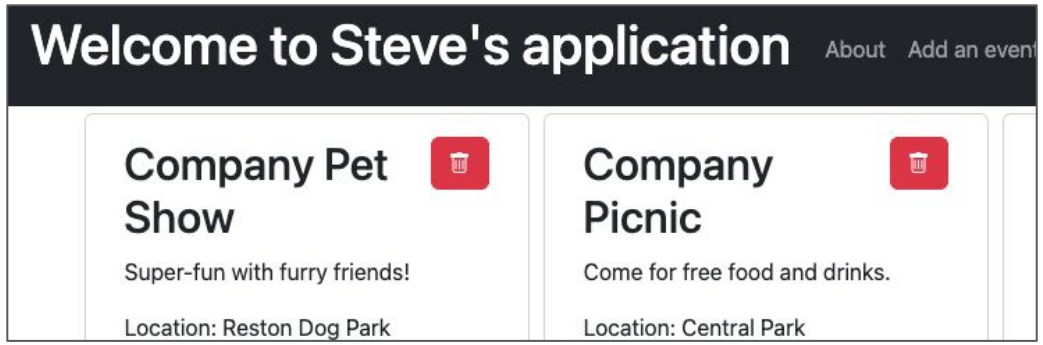
```
$ kubectl get service
NAME             TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)          AGE
events-api-svc   ClusterIP      10.8.10.141   <none>          8082/TCP         3h34m
events-web-svc   LoadBalancer   10.8.13.139   <pending>       80:30062/TCP     3h6m
kubernetes       ClusterIP      10.8.0.1      <none>          443/TCP          2m
```

Accessing the Load Balancer

- Open a new terminal (+ button) and run the following to create the port forward:

```
minikube tunnel & kubectl port-forward service/events-web-svc 8080:80
```

- Click the **Open in Browser** button
- The application should be working again:
  - If you still see Super Mario, just reload the page (the browser cached it)
  - And is now running in Kubernetes!
  - Try adding a few events



- If you see an error on the app home page:
  - The `SERVER` environment variable was not added correctly to the `web-deployment`

Experiment with Replicas

- Scale the `events-web` service:
  - Modify the `web-deployment.yaml` to have `three` replicas
  - Apply the file and test the application
  - Everything should still work fine
- Scale the `events-api` service:
  - Modify the `api-deployment.yaml` to have `three` replicas
  - Apply the file and test the application
  - Now there is a problem with how the events data is stored
  - This is because the `events-api` service is storing state
    - Each copy has its own state
    - This will be corrected when a database is added

Experiment with Replicas (continued)

- Set the replicas back to `1` in both `web-deployment.yaml` and `api-deployment.yaml`
  - Apply both files
  - Verify the pods have scaled back to `1` each

Clean Up

- 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"
    ```

- There is no need to clean up the deployments, services, or pods
  - Leave them running on your cluster

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 created Kubernetes services
  - Created a ClusterIP for the `events-api` service
  - Created a load balancer for the `events-web` service
  - Set environment variables for a Kubernetes deployment