Activity:
Creating a ClusterIP and Load Balancer

Creating the `api-service.yaml`

- In Google Cloud Shell, change into your `kubernetes-config` folder:
  `cd ~/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

Testing a Service DNS Resolution

- You will `exec` into the `events-website` pod and connect to the name of the `events-api-svc` service
  - Retrieve a list of running pods:
    ```
    kubectl get pods
    ```
  - Copy the name of the `events-web` pod
  - Execute into the `events-web` pod
    ```
    kubectl exec -i -t PASTE-POD-NAME -- /bin/bash
    ```
  - Run the following command in the pod:
    ```
    curl http://events-api-svc:8082/events
    ```
  - This should connect to the `events-api-svc` service and return a JSON list of events

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
```

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

Add these three lines

Creating the `web-service.yaml`

- In Cloud Shell, change into your `kubernetes-config` folder:
  `cd ~/eventsapp/kubernetes-config`
- In the `kubernetes-config` folder, create a new file named `web-service.yaml`
  - Copy/paste the contents shown here:

```yaml
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:

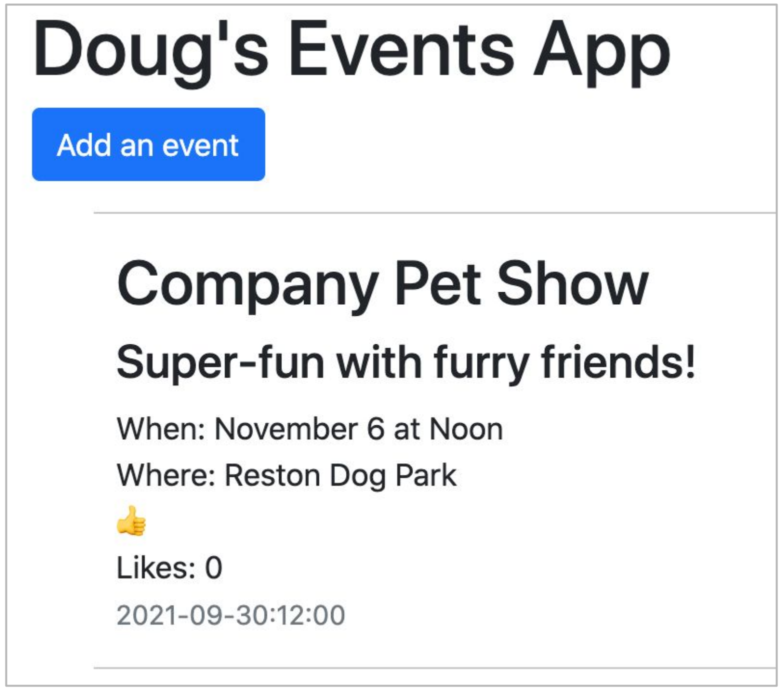
  ```
  kubectl get service
  ```

```
$ 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    34.70.114.185   80:30062/TCP   3h6m
kubernetes       ClusterIP      10.8.0.1       <none>          443/TCP        2m
```

- If you do not have an `EXTERNAL-IP` for the `events-web-svc`, wait a few seconds and try again

Accessing the Load Balancer

- Copy the `EXTERNAL-IP` for the `events-web-svc` and visit it in a new browser tab
- The application should now be working again:
  - The app is now running in Kubernetes!
  - Try adding a few events



Doug's Events App

Company Pet Show

Super-fun with furry friends!

When: November 6 at Noon

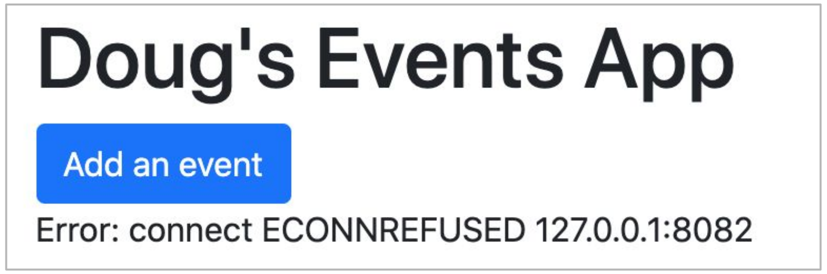
Where: Reston Dog Park

👍

Likes: 0

2021-09-30:12:00

- If you see a connection error as shown here:
  - The `SERVER` environment variable was not added to the `web-deployment`



Doug's Events App

Error: connect ECONNREFUSED 127.0.0.1:8082

Experiment with Replicas

- Scale the `events-web` service:
  - Modify the `web-deployment.yaml` to have `3` 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 `3` 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

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

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