Activity:
Using the `kubectl` CLI

`kubectl` CLI

- From the Google Cloud Console:
  - Click the Navigation menu and select **Kubernetes Engine**
  - On the far right of the cluster table is three vertical dots ⋮
  - Click the three vertical dots, click the **Connect** button, and then choose **Run in Cloud Shell**
    - Cloud Shell will open with a command pre-created
  - Press ENTER to run the command
    - Click **Authorize** if prompted
  - This command creates the `kubectl config` file for your cluster
- The `kubectl config` file is located in the **`.kube`** folder of your home directory
  - Feel free to investigate that file but be sure not to modify it
  - If anything does happen to that file, just run the connect command again

Testing `kubectl` Cluster Access

- 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 three nodes
  - **`kubectl get pods`**
    - This should not return any pods

Deploying a Pod with `kubectl`

- In a Cloud Shell session, make a new folder

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

- In the `kubernetes-config` folder, create a new file named `mario-pod.yaml`
  - You can create the file with the Cloud Shell 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 in cloud shell:

```
cd ~/eventsapp/kubernetes-config/
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
    ```
  - Check the status of the load balancer service:

    ```
    kubectl get service
    ```
  - Keep checking the status until the `EXTERNAL-IP` is populated
- Copy the `EXTERNAL-IP` and try visiting it in a new browser tab
  - You should see the Super Mario game load
    - If you want to play: Press S to start the game and use S to jump

Investigating the Pod

- 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, view running processes in the container
  `ps ax`
- Leave the bash session
  `exit`

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

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