Experiment 2: Docker Commands

1. **Build a Docker Image**:
   Command: `docker build -t [image_name] .`
   Builds a Docker image from a Dockerfile in the current directory.

2. **Run a Docker Container**:
   Command: `docker run -d -p 8080:80 --name [container_name] [image_name]`
   Runs a container in detached mode, mapping port 8080 on your host to port 80 in the container.

3. **List All Containers**:
   Command: `docker ps -a`
   Lists all containers, both running and stopped, on your system.

4. **Stop a Running Container**:
   Command: `docker stop [container_name]`
   Stops a running container by its name.

5. **Remove a Stopped Container**:
   Command: `docker rm [container_name]`
   Removes a stopped container from your system.

6. **Remove a Docker Image**:
   Command: `docker rmi [image_name]`
   Deletes a Docker image from your local machine.

Experiment 3: Create a Docker Image from HTML File

Steps:

1. Create an empty folder hello-container.
2. Open VS Code in that folder.
3. Create index.html file with the content.

```
<style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding-top: 50px;
            background-color: #f0f2f5;
        }
        h1 {
            color: #333;
        }
        h2 {
            color: #666;
        }
    </style>
```

4. Create Dockerfile with the following content.
   # Use the official lightweight Nginx image based on Alpine Linux
   FROM nginx:alpine

   # Remove the default Nginx web content
   RUN rm -rf /usr/share/nginx/html/*
   # Copy our custom index.html into the Nginx web content directory
   COPY index.html /usr/share/nginx/html/
   # Expose port 80 so the container can serve web traffic
   EXPOSE 80

5. Open terminal in VS Code.
6. Run docker --version to check if Docker is running.
   Build the Docker image:
   ```
   docker build -t hello-image .
   ```
7. **Check if the image is created:**
   ```
   docker images
   ```
8. **Create and run the container:**
   ```
   docker run -d -p 8080:80 --name hello-container hello-image
   ```
9. **Check if the container is running:**
   ```
   docker ps -a
   ```
10. **Stop the container:**
   ```
   docker stop hello-container
   ```
11. **Remove the container:**
   ```
   docker rm hello-container
   ```

Experiment 4: Pull Image from Docker Hub

1. Open your terminal in the working directory.

2. Pull the nginx image:
   ```
   docker pull nginx
   ```

3. Verify the image is pulled:
   ```
   docker images
   ```

4. Run the container from the image:
   ```
   docker run -d -p 8080:80 --name [container_name] nginx
   ```

5. Check if the container is running by accessing `localhost:8080`.

6. To stop the container:
   ```
   docker stop [container_name]
   ```

Experiment 5: Create Master and Worker Nodes in Kubernetes

1. Go to KillerCoda and open Kubernetes 1.32.

2. The master and worker nodes will already be created.
   If not created

First, initialize the Kubernetes cluster:

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

Then, set up kubectl so you can run Kubernetes commands:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Finally, check if the cluster is working:

```
kubectl cluster-info
```

3. View the nodes with:
   `kubectl get nodes`

4. The following are key Kubernetes commands:

   - **Get Nodes**:
     `kubectl get nodes`
   - **Get Pods**:
     `kubectl get pods`
   - **Create a Pod**:
     `kubectl run [pod_name] --image=nginx`
   - **Get Pod Details**:
     `kubectl describe pod [pod_name]`
   - **Expose Pod as a Service**:
     `kubectl expose pod [pod_name] --port=80 --type=NodePort`
   - **Get Service**:
     `kubectl get svc`

- **Get Nodes:**
  `kubectl get nodes`
- **Check Client/Server Versions:**
  `kubectl version --short`
- **Get Pods (Current Namespace):**
  `kubectl get pods`
- **Get Pods (Specific Namespace):**
  `kubectl get pods -n [namespace_name]`
- **Get Pod IP Address:**
  `kubectl get pod [pod_name] -o wide`
- **Create a Pod:**
  `kubectl run [pod_name] --image=nginx`
- **Get Pod Details:**
  `kubectl describe pod [pod_name]`
- **Expose Pod as a Service:**
  `kubectl expose pod [pod_name] --port=80 --type=NodePort`
- **Delete a Pod:**
  `kubectl delete pod [pod_name]`

Experiment 6: Create and Troubleshoot a Pod in Kubernetes

1. Go to KillerCoda and open the Kubernetes playground.

2. Create a pod YAML file:
   `nano mypod.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mycontainer
    image: nginx
    ports:
    - containerPort: 80
```

3. Save and exit with:
   `Ctrl + X → Y → Enter`

4. Deploy the pod with:
   `kubectl apply -f mypod.yaml`

5. Verify the pod is running:
   `kubectl get pods`

6. Get the pod's IP address:
   `kubectl get pod mypod -o wide`

7. View the pod logs:
   `kubectl logs mypod`

8. Describe the pod:
   `kubectl describe pod mypod`