**Experiment No. 1**

**AIM:**Create a virtual machine on hypervisor and specify operating system disk size and memory allocation.  
  
**Theory:  
Q1.What is virtualization?**

**Q2.What is a hypervisor?**

**Q3.What are the types of hypervisors?**

**Q4.What is VMWare?**

**Procedure:**Just need to install a VMware and create a new machine with ubuntu iso image and keep the disk size and memory allocation as it is.

**Experiment No. 2**

**AIM:**Run and execute any 6 docker commands and comment on it.

**Theory:**  
**Procedure:**Run any 6 commands provided in the docker resources.pdf.

You can perform these commands after creating an image and running docker.The steps of creating an image,running the container and stopping a container are given in experiment No.3.

**Experiment No. 3  
  
AIM:Create a docker file , docker image and run the container from the image file.**

**Theory:  
Q1.What is containerization?**

**Q2. What is Docker?**

**Q3. What is a docker image?Difference between docker image and container.**

**Q4. State the difference between containerization and virtualization.**

**Q5.What is the purpose of dockerFile?**

**Q6.What are alpine images?**

**Procedure:**Step1: Take any WebL project that has react into it.(I have taken Experiment No.5)  
Step2:Create a Dockerfile into the project structure without any extensions.  
Step3:Include the following code into the Dockerfile:  
# Step 1: Build the React app using Node  
FROM node:18 AS build

# Set working directory inside container  
WORKDIR /app

# Copy dependencies and install them  
COPY package.json package-lock.json ./  
RUN npm install

# Copy the rest of the app  
COPY . .

# Build the app  
RUN npm run build

# Step 2: Serve the build using Nginx  
FROM nginx:alpine

# Copy the build files to nginx's public folder  
COPY --from=build /app/build /usr/share/nginx/html

# Expose port 80  
EXPOSE 80

# Start nginx  
CMD ["nginx", "-g", "daemon off;"]

Step4:Run the following command in the terminal  
docker build -t (name of project) .

Step5:Run the container  
docker run -d -p 3000:80 (name of project)

Step6:You can go to docker desktop and can see your image running.Also to the port where your image is running.

**Experiment No. 4**

**AIM:**Pull an image from a docker hub and perform the following tasks :   
 a) create a container, b) run the container, 3) stop the container  
  
**Theory:  
Q1.What is a docker hub?  
Q2. How do you pull an image from Docker Hub?  
Q3.What happens when you run Docker run?  
Q4.What is the difference between docker create and docker run?  
Q5.What is port mapping in docker?  
Q6.What is the use of -d?**

**Procedure:  
🔹 1. Pull the image from Docker Hub**

**docker pull nginx**

**✔ This command fetches the nginx image from Docker Hub and stores it locally.**

### **🔹 2. Check if the image is pulled**

**docker images**

**✔ This lists all Docker images on your machine. You should see something like:**

**REPOSITORY TAG IMAGE ID CREATED SIZE**

**nginx latest <some-id> <date> 133MB**

### **🔹 3. Create and run a container from the image docker run -d --name my-nginx -p 8080:80 nginx**

* **-d runs the container in detached mode (in the background)**
* **--name my-nginx gives the container a name**
* **-p 8080:80 maps host port 8080 to container port 80**

**✔ Now visit http://localhost:8080 in your browser. You’ll see the Nginx welcome page served from inside the Docker container.**

### **🔹 4. Check running containers docker ps**

**✔ This will show your running container like:**

**CONTAINER ID IMAGE COMMAND STATUS PORTS**

**<id> nginx ... Up 0.0.0.0:8080->80/tcp**

### **🔹 5. Stop the container docker stop my-nginx**

**✔ This will stop the running container named my-nginx.**

### **🔹 6. (Optional) Restart or Remove the container**

**Restart: docker start my-nginx**

**Remove container: docker rm my-nginx**

**Experiment No. 5**

**Aim:**Create a master node and worker node in kubernetes. Run any 6 commands of kubernetes.

**Theory:**

| **Question** | **Answer** |
| --- | --- |
| What is Kubernetes? | Kubernetes is an open-source container orchestration platform that automates deployment, scaling, and management of containerized applications. |
| What is the role of the master node? | The master node controls and manages the Kubernetes cluster by running components like the API server, controller manager, and scheduler. |
| What is a worker node? | A worker node runs containerized applications. It contains the kubelet, kube-proxy, and a container runtime like Docker or containerd. |
| What does kubectl get nodes do? | It lists all the nodes (master and workers) in the cluster. |
| What is a Pod? | A Pod is the smallest deployable unit in Kubernetes, which can hold one or more containers. |
| What is the use of kubectl expose command? | It creates a service to expose a pod or deployment, making it accessible within or outside the cluster. |
| What are the types of services in Kubernetes? | ClusterIP (default), NodePort, LoadBalancer, and ExternalName. |
| Can we run Kubernetes on a local system? | Yes, using tools like Minikube, Kind, or Docker Desktop. However, Killercoda offers an easy cloud-based alternative. |

**Procedure:  
Master node and Worker node are already created in KillerCoda.**

# 1. Check Nodes  
kubectl get nodes

# 2. Create a Pod  
kubectl run nginx-pod --image=nginx

# 3. Get Pod Details  
kubectl get pods  
kubectl describe pod nginx-pod

# 4. Expose Pod as Service  
kubectl expose pod nginx-pod --port=80 --type=NodePort

# 5. Get Services  
kubectl get svc

# 6. Delete Pod  
kubectl delete pod nginx-pod

**Experiment No.6**

**Aim:**Create a pod in kubernetes , find the ip address and do the troubleshooting via log.

Here’s a **step-by-step guide** to create a Pod in Kubernetes, find its IP, and troubleshoot via logs — using **Killercoda**, which is an online Kubernetes playground. I’ll also include **viva questions and answers** you might face.

### **🔧 STEPS TO PERFORM IN KILLERCODA**

#### **Step 1: Launch Killercoda**

* Go to [https://killercoda.com](https://killercoda.com/)
* Choose **"Kubernetes Playground"** → Click **Start Scenario**

#### **Step 2: Create a Pod YAML File**

Use nano or vim to create a pod config:

nano mypod.yaml

Paste this content:

apiVersion: v1

kind: Pod

metadata:

name: myapp-pod

labels:

app: myapp

spec:

containers:

- name: myapp-container

image: nginx

ports:

- containerPort: 80

Save and exit (Ctrl + O, Enter, Ctrl + X).

#### **Step 3: Create the Pod**

kubectl apply -f mypod.yaml

Check pod status:

kubectl get pods

#### **Step 4: Get Pod IP Address**

kubectl get pod myapp-pod -o wide

This shows the Pod IP under the IP column.

#### **Step 5: Troubleshoot Using Logs**

If the pod is running:

kubectl logs myapp-pod

If the pod crashes, you can check logs of the previous container:

kubectl logs myapp-pod --previous

If the pod isn't starting, describe it:

kubectl describe pod myapp-pod

#### **1. Q: What is a Pod in Kubernetes?**

**A:** A Pod is the smallest and simplest unit in Kubernetes. It can contain one or more containers that share storage, networking, and a specification for how to run the containers.

#### **2. Q: How do you create a Pod in Kubernetes?**

**A:** By writing a YAML file describing the Pod and using kubectl apply -f <file> to create it.

#### **3. Q: How do you get the IP address of a Pod?**

**A:** Using kubectl get pod <pod-name> -o wide, which shows the IP address in the IP column.

#### **4. Q: How can you check logs of a Pod?**

**A:** Use kubectl logs <pod-name> to view the stdout of the container running inside the Pod.

#### **5. Q: What if the Pod is crashing or restarting?**

**A:** Use kubectl describe pod <pod-name> to check events and reasons. Also, kubectl logs <pod-name> --previous shows logs from the previous instance.

#### **6. Q: Can a Pod have multiple containers? Why would you do that?**

**A:** Yes, using the containers: array. This is done for tightly coupled containers that need to share resources (e.g., sidecar patterns like logging or monitoring).