**Hands-On Lab: Docker and Kubernetes Advanced Practice**

**1. Docker Basics**

* **Objective:** Gain a comprehensive understanding of Docker commands and their functionality.
* **Steps:**
  + **Installation:**
    - Install Docker on your system (Linux, Windows, or macOS). For Windows and macOS, use Docker Desktop.
    - Verify the installation with docker --version.
  + **Running Containers:**
    - Run the hello-world container to test Docker installation: docker run hello-world.
    - Run a detached container using the nginx image: docker run -d nginx.
    - List all running containers with docker ps.
    - Inspect a container's details: docker inspect <container\_id>.
    - Stop, start, and remove a container using docker stop <container\_id>, docker start <container\_id>, and docker rm <container\_id>.
  + **Managing Containers:**
    - Restart a stopped container using docker restart <container\_id>.
    - Attach to a running container's shell (e.g., nginx): docker exec -it <container\_id> /bin/bash.
    - Remove all stopped containers with docker container prune.

**2. Working with Docker Images**

* **Objective:** Explore Docker images, their layers, and their use in container creation.
* **Steps:**
  + **Pulling Images:**
    - Pull the nginx image from Docker Hub: docker pull nginx.
    - List all downloaded images using docker images.
    - Inspect image details: docker inspect nginx.
  + **Image Layers:**
    - View the layers of the nginx image: docker history nginx.
    - Create a new container from the nginx image: docker run -d nginx.
    - Commit changes made to a running container into a new image: docker commit <container\_id> my\_nginx\_image.
  + **Saving and Loading Images:**
    - Save the nginx image to a tar file: docker save -o nginx.tar nginx.
    - Remove the nginx image from your local machine: docker rmi nginx.
    - Load the saved image from the tar file: docker load -i nginx.tar.

**3. Creating and Sharing Docker Images**

* **Objective:** Build and share custom Docker images.
* **Steps:**
  + **Dockerfile Creation:**
    - Create a directory for your project and navigate to it: mkdir myapp && cd myapp.
    - Write a Dockerfile for a basic Node.js application:

Dockerfile

FROM node:14

WORKDIR /app

COPY package.json ./

RUN npm install

COPY . .

CMD ["node", "app.js"]

EXPOSE 3000

* + - Create a simple app.js file that serves a "Hello World" message.
    - Build the Docker image: docker build -t myapp:1.0 ..
    - Run the image as a container: docker run -d -p 3000:3000 myapp:1.0.
  + **Tagging and Pushing Images:**
    - Tag your image for Docker Hub: docker tag myapp:1.0 <your\_dockerhub\_username>/myapp:1.0.
    - Log in to Docker Hub: docker login.
    - Push your image to Docker Hub: docker push <your\_dockerhub\_username>/myapp:1.0.
  + **Sharing and Pulling:**
    - Share the image URL with a peer and have them pull and run the image.
    - Pull the image on another machine and run it: docker pull <your\_dockerhub\_username>/myapp:1.0.

**4. Working with Dockerfiles**

* **Objective:** Optimize Dockerfiles for efficient image creation.
* **Steps:**
  + **Basic Dockerfile:**
    - Create a basic Dockerfile for a Python Flask application:

Dockerfile

FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY . .

CMD ["python3", "app.py"]

EXPOSE 5000

* + - Build and run the image.
  + **Optimizing the Dockerfile:**
    - **Minimizing Layers:**
      * Combine the COPY and RUN commands to reduce the number of layers.
    - **Multi-Stage Builds:**
      * Implement a multi-stage build to reduce the final image size:

Dockerfile

FROM node:14 as builder

WORKDIR /app

COPY . .

RUN npm install && npm run build

FROM nginx:alpine

COPY --from=builder /app/build /usr/share/nginx/html

* + - **Caching Dependencies:**
      * Reorder Dockerfile commands to leverage Docker's layer caching, ensuring that the RUN command is only re-executed when necessary.
    - **Using a Slim Base Image:**
      * Switch to a lighter base image (e.g., python:3.8-alpine) to reduce the overall image size.
  + **Building and Testing:**
    - Build the optimized image and compare its size with the original.
    - Run the optimized image and ensure it functions correctly.

**5. Docker Hub**

* **Objective:** Explore Docker Hub for image management and collaboration.
* **Steps:**
  + **Docker Hub Account:**
    - Create a Docker Hub account if you don't have one.
    - Log in to Docker Hub from the terminal: docker login.
  + **Exploring Docker Hub:**
    - Search for popular images like mysql, redis, and postgres.
    - View image details, tags, and Dockerfile source.
  + **Image Management:**
    - Push a new version of your image to Docker Hub: docker push <your\_dockerhub\_username>/myapp:1.1.
    - Create a private repository on Docker Hub and push a sensitive image.
    - Test pulling the private image using your credentials on another machine.
  + **Collaborating with Teams:**
    - Create a team on Docker Hub and invite members.
    - Set up access controls for repositories within the team.

**6. Kubernetes Basics**

* **Objective:** Get hands-on experience with Kubernetes, focusing on deployment, scaling, and basic orchestration.
* **Steps:**
  + **Setting Up Kubernetes:**
    - Set up Minikube or use a managed Kubernetes service (GKE, EKS, or AKS).
    - Verify the installation with kubectl version.
  + **Creating a Deployment:**
    - Create a simple deployment using the nginx image:

cmd

kubectl create deployment nginx-deployment --image=nginx

* + - Check the status of the deployment with kubectl get deployments.
    - Describe the deployment to view details: kubectl describe deployment nginx-deployment.
  + **Exposing a Service:**
    - Expose the deployment as a service:

cmd

kubectl expose deployment nginx-deployment --type=NodePort --port=80

* + - Get the service details and access the application via the NodePort.
  + **Scaling the Deployment:**
    - Scale the deployment to 3 replicas:

CMD

kubectl scale deployment nginx-deployment --replicas=3

* + - Verify the scaling with kubectl get pods.
  + **Cleaning Up:**
    - Delete the deployment and service:

CMD

kubectl delete deployment nginx-deployment

kubectl delete service nginx-deployment