Docker Commands and Concepts

Overview

A Docker image is a lightweight, standalone, and executable package that includes everything needed to run an application: code, libraries, dependencies, and system tools. Images are used to create containers, which are instances of images.

Docker images can be stored on Docker Hub or private repositories and are pulled to servers for use. Below is a comprehensive guide to Docker commands and best practices.

Building Docker Images

- **Purpose**: To bundle application code and dependencies into a lightweight image.
- **Command to check available images**:
  docker images

- **Pushing an image to Docker Hub**:
  docker push <image_name>

- **Pulling an image from Docker Hub**:
  docker pull <image_name>

- **Listing available images**:
  docker images

Note: Ensure you are logged into Docker Hub or a private repository with appropriate permissions before pushing or pulling images.

Creating and Managing Containers

- **Run a container from an image**:

  docker run -it <image_name_or_id>
    - -it: Interactive terminal.
- To give the container a specific name:
  docker run -it --name <container_name> <image_name>

● **Exit a container**:

● Regular exit:
exit

● Exit without stopping the container:
CTRL+P + CTRL+Q

● **View running containers**:

docker ps

● **View all containers (running and stopped)**:

docker ps -a

Starting, Stopping, and Removing Containers

● **Start a container**:

docker start <container_name_or_id>

● **Stop a container**:

docker stop <container_name_or_id>

● **Pause a running container**:

docker pause <container_name_or_id>

● **Unpause a paused container**:

docker unpause <container_name_or_id>

● **Remove a container**:

docker rm <container_name_or_id>

● To forcefully remove a running container:
docker rm -f <container_name_or_id>

● **Remove all containers**:

docker rm -f $(docker ps -qa)

Removing Docker Images

● **Remove an image**:
docker rmi <image_name_or_id>

● **Remove all images**:
docker rmi -f $(docker images -q)

Connecting and Executing Commands Inside Containers

● **Reattach to a running container**:
docker attach <container_name_or_id>

● **Execute a command inside a container**:
docker exec -it <container_name_or_id> /bin/bash

Note: Using docker exec ensures the container remains running when you exit.

Converting Containers to Images

● **Create a new image from a container**:
docker commit <container_name_or_id> <new_image_name>

● **Tag an image**:
docker tag <image_name_or_id> <new_tag_name>

Miscellaneous Commands

● **List all container IDs**:
docker ps -aq

● **View container processes**:
ps -ef | grep <container_id>

● **Inspect Docker's home directory**:
cd /var/lib/docker

● **Detach mode (run container in background)**:
docker run -itd <image_name>

● **Rename a container**:
docker rename <old_name> <new_name>

Additional Notes

- Containers are isolated processes managed with namespaces and control groups.
- Applications and dependencies installed within a container remain within the container and do not affect the host system.

To create users inside a container:
useradd <username>