Docker

Sunday, January 28, 2024 8:22 PM

Print

seful commands:
Docker ps
Docker-compose ps
-a flag for all
-Start/stop containers
Start: docker start <container_name_or_id>
Stop: docker stop <container_name_or_id>
Restart: docker restart <container_name_or_id>

-Pull down images from docker hub
docker pull <image_name[:tag]>

-Make new container based on image
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Example: docker run -d --name my_nginx_container -p 8080:80 nginx

-Container modes

Docker containers can be run in different modes to meet various use cases. Here are some common modes of running Docker containers:

- **Detached Mode (`-d`):**
  Running a container in detached mode means it runs in the background. The terminal is not attached to the container, and you can continue using the terminal for other commands. This is often used for long-running services.
  docker run -d --name my_container nginx

- **Interactive Mode (`-it`):**
  Interactive mode allows you to interact with the container's command line. It's commonly used for debugging or exploring the container environment.
  docker run -it --name my_container ubuntu

- **Foreground Mode (Default):**
  If you don't specify **`-d`** or **`-it`**, the container runs in the foreground. The terminal is attached to the container, and you see the container's output. Pressing **`Ctrl+C`** stops the container.
  docker run --name my_container nginx

- **Restart Policies:**
  Docker provides restart policies to automatically restart containers based on different criteria. Common policies include "always," "unless-stopped," and "on-failure."
  docker run --restart always --name my_container nginx

- **Pausing and Resuming:**
  Docker allows you to pause and resume containers. The pause command suspends all processes in the container, and the unpause command resumes them.
  docker pause my_container docker unpause my_container

- **Privileged Mode:**
  Running a container in privileged mode gives it extended privileges on the host. This can be necessary for certain system-level operations but should be used with caution.
  docker run --privileged --name my_container ubuntu

- **Custom Networks:**
  Containers can be connected to custom networks for isolated communication between containers.
  docker network create my_network docker run --network my_network --name container1 nginx docker run --network my_network --name container2 nginx

Print

-What volumes on each container

List all volumes: docker volume ls
Inspect a specific volumes: docker volume inspect <volume_name>
Display only volume names: docker volume ls -q

-Identify docker compose files

The compose file can be put anywhere, but to specify what YAML file to run in docker compose, use: docker-compose -f /path/to/your/docker-compose-file.yml up

If you are already in the directory where your **`docker-compose.yml`** file is located, you can simply run:

```
docker-compose up
docker-compose up
```

To stop the running containers defined in the **`docker-compose.yml`** file, you can use: docker-compose down
Add –v to also remove the volumes.
This command stops and removes the containers, networks, and volumes created by **`docker-compose up`**.

-Basic networking commands (addresses/subnets)

List networks: docker network ls
Inspect network: docker network inspect <network_name_or_id>
Create a network: docker network create <network_name>

Remove a network: docker network rm <network_name_or_id>
Connect container to a network: docker network connect <network_name> <container_name_or_id>
Disconnect container from a network: docker network disconnect <network_name> <container_name_or_id>
Create container with a specific network: docker run --network= <network_name> <other_options> <image>

-Docker compose vs docker (differentiations)

Docker is the engine that runs containers, while Docker Compose is a tool for defining and managing multi-container Docker applications.

Print