Docker Containers and images

CONCEPTS

An image is a read-only template; a container is a running instance of it. Removing a container does not remove its image.

Dockerfile recipe
↓ `docker build`
Image template
↓ `docker run`
Container running instance
↓ `docker push`
Registry Docker Hub

Data written inside a container is lost when it is removed. Keep data in a volume or bind mount.

RUN

`docker run <image>`
Create and start a container

`docker run -d <image>`
Run in the background

`docker run -it <image> sh`
Interactive shell

`docker run --rm <image>`
Delete the container on exit

`docker run --name <name> <image>`
Give the container a name

`docker run -p <host>:<ctr> <image>`
Publish a port

`docker run -v <name>:<path> <image>`
Mount a named volume

`docker run -v <dir>:<path> <image>`
Bind-mount a host folder

`docker run -e KEY=VALUE <image>`
Set an environment variable

CONTAINERS

`docker ps`
List running containers

`docker ps -a`
List all, including stopped

`docker stop <container>`
Stop gracefully

CONTAINERS (cont.)

`docker start <container>`
Start a stopped container

`docker rm <container>`
Remove a stopped container

INSIDE A CONTAINER

`docker exec -it <container> sh`
Open a shell in a running container

`docker exec <container> <cmd>`
Run one command

`docker logs <container>`
Show output

`docker logs -f <container>`
Follow output live

IMAGES

`docker pull <image>`
Download an image

`docker images`
List local images

`docker build -t <name> .`
Build from a Dockerfile here

VOLUMES AND NETWORKS

`docker volume create <name>`
Create a named volume

`docker network create <name>`
Create a bridge network

COMPOSE

`docker compose up -d`
Create and start the stack

`docker compose down`
Stop and remove containers and networks

`docker compose logs -f <service>`
Follow service logs

`docker compose exec <service> sh`
Shell in a service

CLEANUP AND INFO

`docker system prune`
Remove stopped containers and dangling data