# A blue whale with boxes on itA blue whale with boxes on itDocker Commands and Explanations

## About Docker

**Docker:** A platform for developing, shipping, and running applications in containers. It provides tools to create and manage containers efficiently.

**Docker Hub:** A cloud-based service provided by Docker that allows you to store and share Docker images. It's like a marketplace where you can find, use, and distribute container images.

**Docker Repository:** A collection of related Docker images, usually grouped together by a common theme or purpose. Repositories can be public (hosted on Docker Hub) or private (hosted elsewhere).

**Docker Image:** A lightweight, standalone, executable package that contains everything needed to run a piece of software, including the code, runtime, libraries, and dependencies. It's like a snapshot of a containerized application.

**Docker Container:** A runtime instance of a Docker image. It's like a lightweight, isolated virtual environment where your application runs. Containers are portable, scalable, and consistent across different environments.

**Docker Volume:** A persistent data storage mechanism used by Docker containers to store and share data between the host machine and the container. Volumes are used to persist data even after a container is stopped or deleted.

**Docker Port:** Networking mechanism that allows a container to expose specific network ports to the host machine or other containers. It enables communication between containers and external services.

**Docker logs:** View the output and error messages produced by a running container, helping you understand what's happening inside.

**Docker network:** Create and manage networks for containers, allowing them to securely communicate with each other, either on the same host or across different hosts.

**Dockerfile:** A text file that contains instructions for building a Docker image. It defines the environment, dependencies, and configuration of the application inside the container. It's like a recipe for creating Docker images.

**Docker build:** Construct custom container images using a Dockerfile, which contains instructions for building the image, such as installing dependencies and configuring settings.

**Docker Compose:** Simplify the management of multi-container applications by defining them in a single YAML file. It allows you to define the services, networks, and volumes needed for your application and run them together with a single command.

## Docker Images

**docker pull <image-name>:<version>:** This command downloads a specific version of an image from Docker Hub (or another registry). If you don't specify a version, it will default to downloading the latest version.

**docker run -e POSTGRES\_PASSWORD=password -d --name <container-name-to-create> <image-name>:** This command creates and starts a new container from the specified image. -e allows you to set environment variables, -d runs the container in detached mode (in the background), --name assigns a name to the container, and <image-name> specifies the image to use.

**docker run -d -p <host-port>:<container-port> --name <container-name-to-create> <image-name>:** This command creates and starts a new container from the specified image. -p <host-port>:<container-port>: This flag allows you to specify port mapping between the host and the container. It enables communication between the container and the host machine through the specified ports.

**docker images:** This command lists all the images you have downloaded or created on your system.

**docker rmi <image-id>:** This command removes a specific image from your system. You need to specify the ID of the image you want to remove.

## Docker Containers

**docker ps:** This command shows all the currently running containers.

**docker ps -a:** This command shows all containers, including those that are stopped.

**docker rm <container-id>:** This command removes a specific container. You need to specify the ID of the container you want to remove.

**docker start <container-id>:** This command starts a stopped container. You need to specify the ID of the container you want to start.

**docker stop <container-id>:** This command stops a running container. You need to specify the ID of the container you want to stop.

**docker exec -it <container-name> sh:** This command allows you to execute a command inside a running container. -it flags enable an interactive session with the container, meaning you can interact with the command you execute. sh: Specifies the shell (in this case, sh) that you want to run inside the container.

## Docker Logs

**docker logs <container-name>:** This command displays the logs of a specific container. It shows the output that the container has produced.

**docker logs -f <container-name>:** This command allows you to follow (or "tail") the logs of a specific container in real-time.

## Docker Volumes

**docker run -v <volume-name>:<container-path> <image-name>: -v <volume-name>:** This command creates and starts a new Docker container. <container-path>: This option mounts a Docker volume into the container. <volume-name> is the name you give to the volume, and <container-path> is the path within the container where the volume will be mounted.

**docker volume rm <volume-name>:** This command removes one or more Docker volumes.

<volume-name>: Specifies the name of the volume you want to remove.

**docker volume prune:** This command removes all unused Docker volumes.

This is useful for cleaning up unused volumes to reclaim disk space on your system.

**docker volume ls:** This command lists all Docker volumes on your system.

It provides information such as volume name, driver, and mount point.

**docker run -e POSTGRES\_PASSWORD=password -e POSTGRES\_USERNAME=username -p 5432:5432 --name postgres-sql -v C:/Users/lakin/docker/volumes/postgres/data:/var/lib/postgres/data postgres**

* docker run: This command is used to create and start a new Docker container.
* -e POSTGRES\_PASSWORD=password: This option sets an environment variable named POSTGRES\_PASSWORD with the value password. This is used to configure the password for the PostgreSQL database.
* -e POSTGRES\_USERNAME=username: This option sets an environment variable named POSTGRES\_USERNAME with the value username. This is used to configure the username for the PostgreSQL database.
* -p 5432:5432: This option maps port 5432 on the host machine to port 5432 inside the container. Port 5432 is the default port for PostgreSQL, so this allows communication with the PostgreSQL database running inside the container.
* --name postgres-sql: This option assigns the name postgres-sql to the container.
* -v Users/lakin/docker/volumes/postgres/data:/var/lib/postgres/data: This option mounts the host directory Users/lakin/docker/volumes/postgres/data into the container at path /var/lib/postgres/data. This allows persistent storage for the PostgreSQL database data.
* postgres: This is the name of the Docker image from which the container will be created. It specifies that the container will run the PostgreSQL database.

## Docker Build

**docker build -t <image-name-to-create> .**

* docker build: This command builds a Docker image from a Dockerfile and a context (usually the current directory where the Dockerfile is located).
* -t <image-name-to-create>: This option tags the built image with the specified name. The <image-name-to-create> is the name you want to give to the newly created image.
* . This period (.) specifies the build context. It indicates that the Dockerfile and any other files needed for the build are located in the current directory.

## Docker Networks

**docker network ls:** This command lists all Docker networks that are currently created on your system.

**docker network create <network-name-to-create>:** This command creates a new Docker network with the specified name.

**docker network connect <network-name> <container-name-to-connect>:** This command connects a container named <container-name-to-connect> to an existing Docker network named <network-name>. It allows the container to communicate with other containers on that network.

**docker run -d -p <host-port>:<container-port> --name <container-name-to-create> --net <network-name> <image-name>:** This command starts a new container based on the specified <image-name>, and it connects the container to an existing Docker network named <network-name-to-connect>. This enables the container to communicate with other containers on that network.

**docker network inspect <network-name-to-connect>:** This command provides detailed information about the specified Docker network named <network-name-to-connect>. It includes details such as the network's name, ID, driver, and the containers connected to it.

**docker network rm <network-name>:** Deletes a Docker network with the specified name.

## Docker Compose

**docker-compose -p <project-name> up -d:** This is the Docker Compose command-line tool used to manage multi-container Docker applications with a single configuration file (usually named docker-compose.yml). ‘up’ is used to create and start the containers for all services defined in the docker-compose.yml file. It will build images if they don't exist and start the containers for the services.