Docker Hitesh Choudhary

15 September 2024 17:14

What is Images ?

Docker images come from the concept ubuntu guys send disk images sharing the OS and everything you want to run comes in disk.

Images are like disk images which have lots of things on them and we can run them easily .
ubuntu - Official Image | Docker Hub

Docker Desktop | Docker Docs

Docker documentation

https://docs.docker.com/reference/cli/docker/

Docker Commands List

What is a Container ?

The system in which you run the images is known as container.

Docker Commands

```
PS C:\Users\abhik> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx
  compose*    Docker Compose
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container
  desktop*    Docker Desktop commands (Alpha)
  dev*        Docker Dev Environments
  extension*  Manages Docker extensions
  feedback*   Provide feedback, right in your terminal!
  image       Manage images
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Swarm Commands:
  swarm       Manage Swarm

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

Global Options:
      --config string      Location of client config files (default
                           "C:\\Users\\abhik\\.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket to connect to
  -l, --log-level string   Set the logging level ("debug", "info",
                           "warn", "error", "fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\\Users\\abhik\\.docker\\ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "C:\\Users\\abhik\\.docker\\cert.pem")
      --tlskey string      Path to TLS key file (default
                           "C:\\Users\\abhik\\.docker\\key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/
```

docker ps

list all running containers

docker stop

Docker stop container_id

PS C:\Users\abhik> ***docker run --help***

```
Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image

Aliases:
  docker container run, docker run

Options:
      --add-host list                  Add a custom host-to-IP mapping
                                       (host:ip)
      --annotation map                 Add an annotation to the container (passed through to the
                                       OCI runtime) (default map[])
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
      --blkio-weight uint16            Block IO (relative weight), between 10 and 1000, or 0 to
                                       disable (default 0)
      --blkio-weight-device list       Block IO weight (relative device
                                       weight) (default [])
      --cap-add list                   Add Linux capabilities
      --cap-drop list                  Drop Linux capabilities
      --cgroup-parent string           Optional parent cgroup for the
                                       container
      --cgroupns string                Cgroup namespace to use (host|private)
                                       'host':    Run the container in the Docker host's cgroup namespace
                                       'private': Run the container in its own private cgroup namespace
                                       '':        Use the cgroup namespace as configured by the
                                                  default-cgroupns-mode option on the daemon (default)
      --cidfile string                 Write the container ID to the file
      --cpu-period int                 Limit CPU CFS (Completely Fair Scheduler) period
      --cpu-quota int                  Limit CPU CFS (Completely Fair Scheduler) quota
      --cpu-rt-period int              Limit CPU real-time period in microseconds
      --cpu-rt-runtime int             Limit CPU real-time runtime in
                                       microseconds
  -c, --cpu-shares int                 CPU shares (relative weight)
      --cpus decimal                   Number of CPUs
      --cpuset-cpus string             CPUs in which to allow execution
                                       (0-3, 0,1)
      --cpuset-mems string             MEMs in which to allow execution
                                       (0-3, 0,1)
  -d, --detach                         Run container in background and print container ID
      --detach-keys string             Override the key sequence for
                                       detaching a container
      --device list                    Add a host device to the container
      --device-cgroup-rule list        Add a rule to the cgroup allowed
                                       devices list
      --device-read-bps list           Limit read rate (bytes per second) from a device (default [])
      --device-read-iops list          Limit read rate (IO per second)
                                       from a device (default [])
      --device-write-bps list          Limit write rate (bytes per second) to a device (default [])
      --device-write-iops list         Limit write rate (IO per second)
                                       to a device (default [])
      --disable-content-trust          Skip image verification (default
                                       true)
      --dns list                       Set custom DNS servers
      --dns-option list                Set DNS options
      --dns-search list                Set custom DNS search domains
      --domainname string              Container NIS domain name
      --entrypoint string              Overwrite the default ENTRYPOINT
                                       of the image
  -e, --env list                       Set environment variables
      --env-file list                  Read in a file of environment variables
      --expose list                    Expose a port or a range of ports
      --gpus gpu-request               GPU devices to add to the container ('all' to pass all GPUs)
      --group-add list                 Add additional groups to join
      --health-cmd string              Command to run to check health
      --health-interval duration       Time between running the check
                                       (ms|s|m|h) (default 0s)
      --health-retries int             Consecutive failures needed to report unhealthy
      --health-start-interval duration Time between running the check
                                       during the start period
```

Docker command to run different version of postgres

```
PS C:\Users\abhik> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

PS C:\Users\abhik> ***docker run --name postgres-latest -e POSTGRES_PASSWORD=my secretpassword -d postgres //for latest***

```
8d3462273d419626479e114b4e1f033c99b6417f8a3a597c5652f0575d079294
```

- PS C:\Users\abhik> ***docker run --name postgres-old -e POSTGRES_PASSWORD=mysecretpassword -d postgres:13.8 //for old***

```
Unable to find image 'postgres:13.8' locally
13.8: Pulling from library/postgres
e9995326b091: Pull complete
a0cb03f17886: Pull complete
bb26f7e78134: Pull complete
c8e073b7ae91: Pull complete
99b5b1679915: Pull complete
55c520fc03c5: Pull complete
d0ac84d6672c: Pull complete
4effb95d5849: Pull complete
97fd2548fc1e: Pull complete
43e7f13e3769: Pull complete
2898936d5b2e: Pull complete
b4b731b0864d: Pull complete
fbd79522dd4c: Pull complete
Digest: sha256:2b31dc28ab2a687bb191e66e69c2534c9c74107ddb3192ff22a04de386425905
Status: Downloaded newer image for postgres:13.8
37b0c9968fb0d280335e905fd034a6061ee8e35c348b47b8d3b4daebd38e04bf
```

```
PS C:\Users\abhik> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STAT
US          PORTS      NAMES
37b0c9968fb0   postgres:13.8   "docker-entrypoint.s…"   2 minutes ago    Up 2
 minutes   5432/tcp   postgres-old
8d3462273d41   postgres        "docker-entrypoint.s…"   4 minutes ago    Up 4
 minutes   5432/tcp   postgres-latest
```

Running on same port causes an issue hence we need to learn port mapping.

- Docker only runs the application over the system it does not run the entire kernel on top of another OS to run the application this is the difference between docker and VM
- Docker is comprise of App with bare min OS layer to run it.
- Docker container prune to remove all the container

Volume keeps the image information

Port Mapping

---

docker run --name my-mongodb -p 4000:27017 mongo

- -p means the port mapping method

Opening up a port on my machine in compare to what port is mongo is giving to me and connecting both the ports.

docker logs my-mongodb-one -t

Gives details about the container specified here.

To connect one container with another container

Goal mongo-express with mongodb

- docker network create mongo-network
- docker run --name my-mongo-one -e MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=password --net mongo-network -p 27017:27017 -d mongo

Docker Compose

First write docker version
Then services(Images)

```
version: "3"
services:
  mongodb:
  mongo-express:
```

Like this

Docker compose automatically creates a network

```
version: "3"
services:
  mongodb:
    image: mongo
    ports:
      - "27017:27017"
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=paswoord
  mongo-express:
    image: mongo-express
    ports:
      - "8081:8081"
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
      - ME_CONFIG_MONGODB_ADMINPASSWORD=paswoord
      - ME_CONFIG_MONGODB_SERVER=mongodb

to run this docker compose we need the command
Docker-compose -f docker-compose.yaml up //to make it up and running
```

Now docker is sometimes exiting the mongo-express server the reason behind it is that it does not know which image to run first hence we need to specify which to run first.

```
version: "3"
services:
  mongodb:
    image: mongo
    ports:
      - "27017:27017"
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=paswoord
```

Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/

To learn about docker learn the api refferences

Docker pull

Is grabbing the image from the repo

docker images are composed of layer and made it super smart
It is composed of variety of layer.

How to see the images commands

Docker image [command]

```
PS C:\Users\abhik> docker image

Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Download an image from a registry
  push        Upload an image to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
```

postgres - Official Image | Docker Hub
every image details how to run it is given below that
the image details

```
$ docker run --name some-postgres -e POSTGRES_PASSWORD=mysecretpassword -d postgres
```

-e is environment value
-d is for detach mode for keeping terminal free

```
check
                                  (ms|s|m|h) (default 0s)
      --health-retries int        Consecutive failures needed to
                                  report unhealthy
      --health-start-interval duration   Time between running the check
                                  during the start period
                                  (ms|s|m|h) (default 0s)
      --health-start-period duration    Start period for the container
                                  to initialize before starting
                                  health-retries countdown
                                  (ms|s|m|h) (default 0s)
      --health-timeout duration   Maximum time to allow one check
                                  to run (ms|s|m|h) (default 0s)
      --help                      Print usage
  -h, --hostname string           Container host name
      --init                      Run an init inside the container
                                  that forwards signals and reaps
                                  processes
  -i, --interactive               Keep STDIN open even if not attached
      --ip string                 IPv4 address (e.g., 172.30.100.104)
      --ip6 string                IPv6 address (e.g., 2001:db8::33)
      --ipc string                IPC mode to use
      --isolation string          Container isolation technology
      --kernel-memory bytes       Kernel memory limit
  -l, --label list                Set meta data on a container
      --label-file list           Read in a line delimited file of
                                  labels
      --link list                 Add link to another container
      --link-local-ip list        Container IPv4/IPv6 link-local
                                  addresses
      --log-driver string         Logging driver for the container
      --log-opt list              Log driver options
      --mac-address string        Container MAC address (e.g.,
                                  92:d0:c6:0a:29:33)
  -m, --memory bytes              Memory limit
      --memory-reservation bytes  Memory soft limit
      --memory-swap bytes         Swap limit equal to memory plus
                                  swap: '-1' to enable unlimited swap
      --memory-swappiness int     Tune container memory swappiness
                                  (0 to 100) (default -1)
      --mount mount               Attach a filesystem mount to the
                                  container
      --name string               Assign a name to the container
      --network network           Connect a container to a network
      --network-alias list        Add network-scoped alias for the
                                  container
      --no-healthcheck            Disable any container-specified
                                  HEALTHCHECK
      --oom-kill-disable          Disable OOM Killer
      --oom-score-adj int         Tune host's OOM preferences
                                  (-1000 to 1000)
      --pid string                PID namespace to use
      --pids-limit int            Tune container pids limit (set
                                  -1 for unlimited)
      --platform string           Set platform if server is
                                  multi-platform capable
      --privileged                Give extended privileges to this
                                  container
  -p, --publish list              Publish a container's port(s) to
                                  the host
  -P, --publish-all               Publish all exposed ports to
                                  random ports
      --pull string               Pull image before running
                                  ("always", "missing", "never")
                                  (default "missing")
  -q, --quiet                     Suppress the pull output
      --read-only                 Mount the container's root
                                  filesystem as read only
      --restart string            Restart policy to apply when a
                                  container exits (default "no")
      --rm                        Automatically remove the
                                  container and its associated
                                  anonymous volumes when it exits
      --runtime string            Runtime to use for this container
      --security-opt list         Security Options
      --shm-size bytes            Size of /dev/shm
      --sig-proxy                 Proxy received signals to the
                                  process (default true)
      --stop-signal string        Signal to stop the container
      --stop-timeout int          Timeout (in seconds) to stop a
                                  container
      --storage-opt list          Storage driver options for the
                                  container
      --sysctl map                Sysctl options (default map[])
      --tmpfs list                Mount a tmpfs directory
  -t, --tty                       Allocate a pseudo-TTY
      --ulimit ulimit             Ulimit options (default [])
  -u, --user string               Username or UID (format:
                                  <name|uid>[:<group|gid>])
      --userns string             User namespace to use
      --uts string                UTS namespace to use
  -v, --volume list               Bind mount a volume
      --volume-driver string      Optional volume driver for the
                                  container
      --volumes-from list         Mount volumes from the specified
                                  container(s)
  -w, --workdir string            Working directory inside the
                                  container
```

```yaml
  mongodb:
    image: mongo
    ports:
      - "27017:27017"
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=paswoord
  mongo-express:
    image: mongo-express
    restart: always //here restart means container will restart until it's find a connection
    ports:
      - "8081:8081"
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
      - ME_CONFIG_MONGODB_ADMINPASSWORD=paswoord
      - ME_CONFIG_MONGODB_SERVER=mongodb
```

In major time the data does not persist in the docker means the container is detachable. It is good for core system what if we want to persist this data if we want to keep the data we use volume.

```yaml
version: "3"
services:
  mongodb:
    image: mongo
    ports:
      - "27017:27017"
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=paswoord
    volumes:
      - my-mongo-data:/data/db
  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - "8081:8081"
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
      - ME_CONFIG_MONGODB_ADMINPASSWORD=paswoord
      - ME_CONFIG_MONGODB_SERVER=mongodb
volumes:
  my-mongo-data:
    driver: local
```

In this we predefine the volumes which we are keeping locally and we are setting up the path for mongo-db in the mongodb service and then the container path which is predefined path we need to see for every container we do so.

Now, if we change this volume name then we can create multiple volumes can be created for a database.

```
┌──(abhik㉿Abhik)-[/mnt/d/Erricson/Docker]
└─$ docker-compose -f docker-compose.yaml down
WARN[0000] /mnt/d/Erricson/Docker/docker-compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
 ✔ Container docker-mongodb-1        Removed     0.0s
 ✔ Container docker-mongo-express-1  Removed     0.0s
 ✔ Network docker_default            Removed     0.2s
```

```dockerfile
FROM python:3-alpine3.15
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 3000
CMD python ./index.py
```

FORM means the base over which it is gonna run
WORKDIR is working directory it is gonna create
COPY means copying the code to the work dir
RUN is the running the base installations
EXPOSE means which port we are gonna use
And CMD is the command to run the script.

docker build -t abhik2004/hey-python-flask:0.0.1.RELEASE .

Command to build the image last . Means the files are in current directory.

```
┌──(abhik㉿Abhik)-[/mnt/d/Erricson/Docker/python]
└─$ docker container run -p 4000:3000 abhik2004/hey-python-flask:0.0.1.RELEASE
```

First port is local port and second is the described port

```
┌──(abhik㉿Abhik)-[/mnt/d/Erricson/Docker/python]
└─$ docker push abhik2004/hey-python-flask:0.0.1.RELEASE
```

Pushing the image to the docker repo

For Node JS

```json
{
  "name": "docker",
  "version": "1.0.0",
  "description": "Learning docker ",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "node index.js"
  },
  "author": "Abhik Ghosh",
  "license": "ISC",
  "dependencies": {
    "express": "^4.21.0"
  }
}
```

Change the package.json script from test to start