Docker Notes

Created By:
Adipta Basu

Docker

Docker is an open-source platform that automates the deployment and management of applications inside containers. Containers are lightweight, portable, and self-sufficient environments that package everything needed to run an application, including code, runtime, libraries, and dependencies.

Basic Docker Concepts

1. Containers: Docker uses containers to encapsulate applications and their dependencies. Containers are isolated, efficient, and consistent across different environments.
2. Image-based: Docker uses images as the building blocks for containers. Images are read-only templates that contain all the necessary instructions to create a container.
3. Portability: Docker containers can run on any machine that has Docker installed, regardless of the underlying operating system or infrastructure.
4. Microservices: Docker facilitates the adoption of microservices architecture by allowing applications to be broken down into smaller, loosely-coupled components that can be managed and scaled independently.
5. DevOps: Docker promotes DevOps practices by enabling developers to build, test, and deploy applications quickly and consistently across development, testing, and production environments.
6. **Orchestration:** Docker can be integrated with orchestration tools like Kubernetes or Docker Swarm to automate container deployment, scaling, and management.

Docker Commands

1. Docker Version and Information

- docker version

  This command shows the docker version and information.

- docker info

  This command displays the system wide information.

- docker login [registry]

  This command helps log into a Docker Registry.

- docker logout [registry]

  This command helps logout from an existing Docker Registry.

2. Docker System Commands

- docker system df

  This command displays a summary of the disk usage for Docker Objects including images, containers, and volumes.

  - '-v' → Adding this option to the command shows the detailed information on disk space usage.

- docker system events

  This command streams real time events from the docker daemon.

  - OPTIONS:

    - '--since' → Adding this option to the command shows all events since a given timestamp or relative time.

      Example:

    - '--untill' → Adding this option to the command streams events untill a given timestamp or relative time.

      Example:

- docker system info

  This command displays system-wide information regarding the Docker, and its resources, such as number of containers images, driver information, system information, system status, etc.

- docker system prune

  This command is used to remove unused data.

  - OPTIONS:

    - '-a', '--all' → Remove all unused images, not just dangling ones.
    - '--volumes' → Removes volumes along with other types of data.
    - '-f', '--force' → Do not prompt for confirmation.

3. Using Docker Images

- `docker images`

  Lists all local images.

- `docker pull [image]`

  Pulls an image from the registry.

- `docker push [username/image]`

  Push an image to the registry.

- `docker build -t [tag]`

  Build an image from the Dockerfile in the current directory.

- `docker rmi`

  Removes an image.

4. Managing Containers

- docker run [options] [image]

  Start Container

  - OPTIONS:
    - '-d' → Run containers in the background.
    - '--name' → Assign a name to the container.
    - '-p [host port]:[container port]' → Maps the port on the container to the port on the host.
    - '-v [host dir]:[container dir]' → Bind mount a volume.

- docker ps

  Lists all the running containers.

  - OPTIONS:
    - '-a' → Lists all the containers, not just the running ones.

- docker stop [container]

  Stops a running container.

- docker start [container]

  Starts a stopped container.

- docker restart [container]

  Restart a container.

- docker rm [container]

  Remove a stopped container.

- docker logs [container]

  Fetches the logs.

  - OPTIONS:
    - '-f' → Keeps the session active, and keeps fetching the logs.

- docker exec -it [container] [command]

  Execute a command inside a running container.

- docker inspect [object name]

  Return low-level information on Docker objects.

5. Docker Networking

- docker network create [options] [name]

  Creates a new network.

  - OPTIONS:
    - '--driver', '-d' → Specify the network driver. Common drivers are 'bridge', 'overlay', 'macvlan'. The default is bridge.
    - '--subnet' → Assign a specific IP subnet in CIDR Format, such as '192.168.1.0/24'.
    - '--ip-range' → Allocate a specific IP Range within the subnet, which limits IPs assigned to containers.
    - '--gateway' → Specify a custom network gateway for the subnet.
    - '--ipam-driver' → Define an IP Address Management (IPAM) driver to be used. The default is 'default', which is Docker built-in IPAM driver.
    - '--ipam-opt' → Set options for the IPAM driver as key-value pairs.
    - '--opt' → Set driver specific options using key-value pairs.
    - '--attachable' → Allows manual container attachment.
    - '--internal' → Restricts external access to and from the network.
    - '-label' → Add metadata to the network in the form of key-value pairs.

Creating a bridge network with a specified subnet and gateway.

```
docker network create --driver bridge --subnet 192.168.100.0/24 --gateway 192.168.100.1 my_network
```

Used in Docker Swarm to connect multiple Docker daemons together and enable swarm service to communicate with each other. You can specify options like '--attachable' to allow containers to connect to the overlay network.

```
docker network create --driver overlay --attachable my_network
```

Allows you to assign a MAC Address to a container making it appear like a physical device.

```
docker network create --driver macvlan --subnet 192.168.200.0/24 --gateway 192.168.200.1 -o parent=eth0 my_network
```

- `docker network ls`

  List all networks.

- `docker network rm [network]`

  Removes networks.

6. Docker Volumes

- docker volume create [options] [volume]

  Creates a new volume.

  - OPTIONS:

    - '--driver' → Specify the volume driver to use. The default driver is 'Local'.
      Example → docker volume create --driver local my_volume

    - '--label' → Add metadata to the volume in the form of key-value pairs.
      Example → docker volume create --label environment=production --label team=devops my-vol

    - '--opt' → Specify options for the volume driver. Options vary depending on the driver used.

- docker volume ls

  Lists all volumes

- docker volume inspect [volume name]

  Displays detailed information about a volume.

- docker volume rm [volume name]

  Remove a volume.

- docker volume prune

  Remove all unused volumes.

-
  ```
  docker run -v [volume name]:[path in container] [options] [image name]
  ```

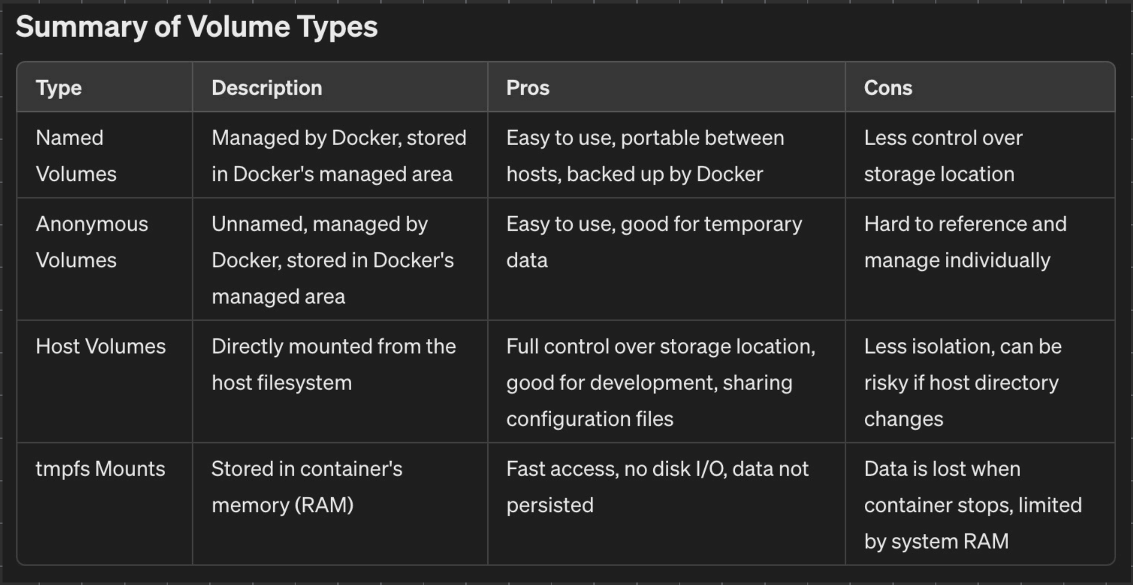
  Runs a container and mounts a volume to a specified path inside the container.

  ```
  docker run -d -v my_volume:/data my_image
  ```

-
  ```
  docker run -v [path on host]:[path in container] [options] [image name]
  ```

  Runs a container and mounts a host directory as a volume.

  ```
  docker run -d -v /host/data:/container/data my_image
  ```



Summary of Volume Types

Type	Description	Pros	Cons
Named Volumes	Managed by Docker, stored in Docker's managed area	Easy to use, portable between hosts, backed up by Docker	Less control over storage location
Anonymous Volumes	Unnamed, managed by Docker, stored in Docker's managed area	Easy to use, good for temporary data	Hard to reference and manage individually
Host Volumes	Directly mounted from the host filesystem	Full control over storage location, good for development, sharing configuration files	Less isolation, can be risky if host directory changes
tmpfs Mounts	Stored in container's memory (RAM)	Fast access, no disk I/O, data not persisted	Data is lost when container stops, limited by system RAM