

All Docker Commands

Here's a comprehensive list of commonly used Docker commands, along with their usage:

Basic Docker Commands

1. **docker version**: Displays Docker version information.
2. **docker info**: Provides detailed information about the Docker installation.
3. **docker --help**: Lists all available Docker commands and options.

Image Management Commands

1. **docker pull <image>**: Downloads an image from a Docker registry (e.g., Docker Hub).
 - Example: `docker pull nginx`
2. **docker images**: Lists all Docker images available on the system.
 - Example: `docker images`
3. **docker rmi <image>**: Deletes a Docker image from the system.
 - Example: `docker rmi nginx`
4. **docker build -t <name> <path>**: Builds a Docker image from a Dockerfile.
 - Example: `docker build -t myapp:latest .`
5. **docker tag <source_image> <target_image>**: Tags an image with a new name.
 - Example: `docker tag nginx:latest myrepo/nginx:v1`
6. **docker save -o <file> <image>**: Saves an image to a tar archive.
 - Example: `docker save -o nginx.tar nginx:latest`
7. **docker load -i <file>**: Loads an image from a tar archive.
 - Example: `docker load -i nginx.tar`

Container Management Commands

1. **docker run <image>**: Creates and starts a new container from an image.
 - Example: `docker run nginx`
2. **docker run -d <image>**: Runs a container in detached mode (in the background).
 - Example: `docker run -d nginx`
3. **docker run -it <image>**: Runs a container interactively with a terminal.
 - Example: `docker run -it ubuntu bash`
4. **docker ps**: Lists all running containers.
 - Example: `docker ps`
5. **docker ps -a**: Lists all containers, including stopped ones.

- Example: `docker ps -a`
- 6. **docker stop <container>**: Stops a running container.
 - Example: `docker stop my_container`
- 7. **docker start <container>**: Starts a stopped container.
 - Example: `docker start my_container`
- 8. **docker restart <container>**: Restarts a container.
 - Example: `docker restart my_container`
- 9. **docker rm <container>**: Deletes a stopped container.
 - Example: `docker rm my_container`
- 10. **docker exec -it <container> <command>**: Executes a command in a running container.
 - Example: `docker exec -it my_container bash`
- 11. **docker logs <container>**: Displays logs from a container.
 - Example: `docker logs my_container`
- 12. **docker attach <container>**: Attaches to a running container's console.
 - Example: `docker attach my_container`
- 13. **docker kill <container>**: Forcefully stops a container.
 - Example: `docker kill my_container`

Container Networking Commands

1. **docker network ls**: Lists all Docker networks.
 - Example: `docker network ls`
2. **docker network create <name>**: Creates a new Docker network.
 - Example: `docker network create my_network`
3. **docker network rm <name>**: Deletes a Docker network.
 - Example: `docker network rm my_network`
4. **docker network connect <network> <container>**: Connects a container to a network.
 - Example: `docker network connect my_network my_container`
5. **docker network disconnect <network> <container>**: Disconnects a container from a network.
 - Example: `docker network disconnect my_network my_container`

Volume Management Commands

1. **docker volume ls**: Lists all Docker volumes.
 - Example: `docker volume ls`
2. **docker volume create <name>**: Creates a new Docker volume.
 - Example: `docker volume create my_volume`
3. **docker volume rm <name>**: Deletes a Docker volume.
 - Example: `docker volume rm my_volume`
4. **docker volume inspect <name>**: Displays detailed information about a volume.

- Example: `docker volume inspect my_volume`
- 5. **docker run -v <volume>:/path/in/container <image>**: Mounts a volume into a container.
 - Example: `docker run -v my_volume:/data nginx`

Dockerfile Commands

1. **docker build -f <Dockerfile>**: Builds an image from a specific Dockerfile.
 - Example: `docker build -f Dockerfile .`

Docker Compose Commands

1. **docker-compose up**: Starts containers defined in a `docker-compose.yml` file.
 - Example: `docker-compose up`
2. **docker-compose down**: Stops and removes containers, networks, and volumes created by `docker-compose up`.
 - Example: `docker-compose down`
3. **docker-compose ps**: Lists containers created by Docker Compose.
 - Example: `docker-compose ps`
4. **docker-compose logs**: Shows logs for containers managed by Docker Compose.
 - Example: `docker-compose logs`
5. **docker-compose build**: Builds or rebuilds services defined in a Compose file.
 - Example: `docker-compose build`

Image and Container Inspection

1. **docker inspect <container_or_image>**: Returns low-level information about a container or image.
 - Example: `docker inspect my_container`
2. **docker top <container>**: Displays running processes in a container.
 - Example: `docker top my_container`
3. **docker stats**: Displays resource usage statistics of running containers.
 - Example: `docker stats`

System Cleanup Commands

1. **docker system df**: Displays information about disk usage by Docker.
 - Example: `docker system df`
2. **docker system prune**: Removes unused data (stopped containers, unused networks, dangling images).
 - Example: `docker system prune`
3. **docker image prune**: Removes unused and dangling images.
 - Example: `docker image prune`

4. **docker container prune**: Removes all stopped containers.
 - Example: `docker container prune`

Other Commands

1. **docker commit <container> <image>**: Creates a new image from a container's changes.
 - Example: `docker commit my_container my_image`
2. **docker export <container>**: Exports a container's filesystem to a tar archive.
 - Example: `docker export my_container > container.tar`
3. **docker import <file>**: Imports a tarball to create an image.
 - Example: `docker import container.tar my_imported_image`

These Docker commands cover the most common activities when working with Docker, ranging from managing containers, images, and volumes to orchestrating multi-container applications with Docker Compose. Mastering these commands helps in efficiently creating, deploying, and managing containerized applications.