Author: Mahesh Babu

Title: Docker Command Reference Guide

Subject: Docker, DevOps

Docker Command Reference Guide

Docker is a powerful platform for developing, shipping, and running applications in containers. Below is a comprehensive guide to the most commonly used Docker commands, organized by functionality for efficient workflow management.

Managing Docker Images

Docker images are the core components used to create containers. Proper management of these images is crucial for maintaining an efficient and streamlined development environment.

1. Image Retrieval and Management

- Pull an Image from Docker Hub:

  - Command: `docker pull <image_name>`

  - Usage: Downloads the specified image from Docker Hub or any configured registry to your local machine.

- Search for Images on Docker Hub:

  - Command: `docker search <image_name>`

- Usage: Searches Docker Hub for images matching the specified name or keywords.

- List Downloaded Images:

- Command: `docker images` or `docker image ls`

- Usage: Displays all images stored locally, including their IDs, repositories, tags, and sizes.

2. Image Creation and Customization

- Build an Image from a Dockerfile:

- Command: `docker build -t <image_name> .`

- Usage: Compiles an image from a Dockerfile in the current directory, tagging it with a specified name.

- Create an Image from a Running Container:

- Command: `docker commit <container_id/container_name> <image_name>`

- Usage: Captures the current state of a container as a new image.

3. Image Inspection and Maintenance

- Get Detailed Information about an Image:

- Command: `docker image inspect <image_name/image_id>`

- Usage: Returns low-level metadata about the image, including its layers, creation date, and configuration.

- Save an Image as a Tarball:

- Command: `docker save -o <tarfile_name> <image_name>`

- Usage: Exports the image as a tar file for backup or transfer.

- Load an Image from a Tarball:

- Command: `docker load -i <tarfile_name>`

- Usage: Imports an image from a tar file into the local Docker repository.

4. Image Cleanup and Optimization

- Delete an Image Not Linked to Any Container:

  - Command: `docker rmi <image_name/image_id>`

  - Usage: Removes a specified image if no container is using it, freeing up disk space.

- Force Delete an Image Linked to a Container:

  - Command: `docker rmi -f <image_name/image_id>`

  - Usage: Forcefully removes an image, even if containers are dependent on it.

- Delete All Unused Images:

  - Command: `docker system prune -af`

  - Usage: Removes all unused images, containers, volumes, and networks, optimizing disk usage.

Managing Docker Containers

Containers are the running instances of Docker images. Effective container management ensures smooth application deployment and operation.

1. Container Lifecycle Management

- List Running Containers:

  - Command: `docker container ls`

  - Usage: Displays all currently running containers, including their IDs, names, and resource usage.

- List All Containers (Running and Stopped):

  - Command: `docker ps -a`

- Usage: Shows all containers, whether running, stopped, or exited, along with their statuses.

- Start a Container:

- Command: `docker start <container_id/container_name>`

- Usage: Initiates a previously stopped container without altering its configuration.

- Stop a Container:

- Command: `docker stop <container_id/container_name>`

- Usage: Gracefully stops a running container by sending a SIGTERM signal.

- Restart a Container:

- Command: `docker restart <container_id/container_name>`

- Usage: Stops and immediately restarts a container, useful for applying updates or changes.

- Example with Delay: `docker restart -t 10 <container_id/container_name>` (Restarts after a 10-second delay).

2. Container Cleanup and Resource Management

- Remove a Stopped Container:

- Command: `docker rm <container_id/container_name>`

- Usage: Deletes a container that is no longer running, freeing up its resources.

- Force Remove a Running Container.

- Command: `docker rm -f <container_id/container_name>`

- Usage: Immediately stops and deletes a container, potentially disrupting services.

- Stop All Running Containers:

- Command: `docker stop $(docker ps -aq)`

- Usage: Stops every running container, useful for system maintenance.

- Remove All Stopped Containers:

- Command: `docker rm $(docker ps -aq)`

- Usage: Cleans up all containers that are not currently running.

- Remove All Containers (Running and Stopped):

- Command: `docker rm -f $(docker ps -aq)`

- Usage: Purges all containers from the system, regardless of their state.

3. Container Inspection and Monitoring

- Inspect Container Details:

- Command: `docker inspect <container_id/container_name>`

- Usage: Provides detailed information about the container's configuration and state.

- View Container Logs:

- Command: `docker logs <container_id/container_name>`

- Usage: Outputs the logs generated by the container's processes, essential for troubleshooting.

4. Advanced Container Operation.

- Run a Container with Options:

- Command: `docker run <image_name/image_id> [OPTIONS]`

- Usage: Starts a new container from an image, with a variety of options to customize its operation.

- Common Options:

- `--name <name>`: Assigns a custom name to the container.

- `-d`: Runs the container in detached mode (background).

- `-it`: Opens an interactive terminal inside the container.

- `-p <host_port>:<container_port>`: Maps host ports to container ports.

- `-e <env_var>=<value>`: Passes environment variables to the container.

- `-v <host_path>:<container_path>`: Mounts a volume inside the container.

- `--network <network_name>`: Connects the container to a specific network.

- `--rm`: Automatically removes the container when it exits.

- `-m <memory_limit>`: Limits the memory usage of the container.

- Execute a Command Inside a Running Container:

  - Command: `docker exec -it <container_id/container_name> <command>`

  - Usage: Runs a specific command inside an active container, such as starting a shell or script.

- Detach from a Container Without Stopping It:

  - Command: `Ctrl + p, Ctrl + q`

  - Usage: Detaches from an interactive session, leaving the container running in the background.

- Reattach to a Running Container:

  - Command: `docker attach <container_id/container_name>`

  - Usage: Reopens an interactive terminal session with a running container.

5. Container Networking

- View Ports Used by a Container:

  - Command: `docker port <container_id/container_name>`

  - Usage: Lists the port mappings between the container and the host.

- View Processes Running in a Container:

  - Command: `docker top <container_id/container_name>`

  - Usage: Displays the active processes running inside the container, similar to the `top` command in Linux.

Managing Docker Networks

Docker networks enable containers to communicate with each other and with external systems. Proper network management ensures secure and efficient communication.

1. Network Creation and Management

- List Docker Networks:

  - Command: `docker network ls`

  - Usage: Displays all available Docker networks, including their IDs, drivers, and scopes.

- Create a Docker Network:

  - Command: `docker network create --driver <network_driver> <network_name>`

  - Usage: Establishes a new network with a specified driver, such as `bridge`, `overlay`, or `host`.

2. Network Inspection and Maintenance

- Inspect Network Details:

  - Command: `docker network inspect <network_name/network_id>`

  - Usage: Provides detailed information about the network, including connected containers and settings.

- Delete a Docker Network:

  - Command: `docker network rm <network_name/network_id>`

  - Usage: Removes a specified network, provided no containers are connected to it.

3. Connecting and Disconnecting Containers

- Connect a Running Container to a Network:

- Command: `docker network connect <network_name/network_id> <container_name/container_id>`

- Usage: Links an existing container to a specified network without restarting it.

- Disconnect a Running Container from a Network:

- Command: `docker network disconnect <network_name/network_id> <container_name/container_id>`

- Usage: Detaches a container from a network while keeping it running.

Managing Docker Volumes

Docker volumes are used to persist data generated by and used by Docker containers. Managing these volumes efficiently is key to data integrity and availability.

1. Volume Creation and Management

- List Docker Volumes:

- Command: `docker volume ls`

- Usage: Displays all volumes created in Docker, including their names and drivers.

- Create a Docker Volume:

- Command: `docker

volume create <volume_name>`

- Usage: Creates a new volume that can be attached to one or more containers for persistent storage.

2. Volume Inspection and Maintenance

- Inspect Volume Details:

  - Command: `docker volume inspect <volume_name/volume_id>`

  - Usage: Provides detailed information about the volume, including its mount point and associated containers.

- Delete a Docker Volume:

  - Command: `docker volume rm <volume_name/volume_id>`

  - Usage: Removes a volume, provided it is not currently in use by any container.

This guide is designed to serve as a comprehensive reference for both beginners and experienced professionals working with Docker. By following these best practices and commands, you can efficiently manage your Docker environment, ensuring robust, scalable, and reliable containerized applications.