



CDW Documentation

Docker Basics - Command Reference

Docker Basics - Command Reference

Docker Installation and Health Checks

docker version

Purpose

Verifies that the Docker CLI is installed and whether it can communicate with the Docker daemon.

What it shows

Client version information and server (daemon) version information.

Usage

```
docker version
```

Notes

If the Docker daemon is not running, only client information is shown and the server section fails.

docker info

Purpose

Displays detailed system-wide Docker configuration and runtime information.

What it shows

Docker root directory, storage driver, number of images and containers, OS and architecture details.

Usage

```
docker info
```

Working With Containers

docker run

Purpose

Creates and starts a new container from an image.

Common usage

```
docker run hello-world
```

```
docker run -it ubuntu bash
```

```
docker run -d --name my-nginx -p 8080:80 nginx
```

Key options

-it runs the container interactively

-d runs the container in the background

--name assigns a readable name to the container

-p maps host ports to container ports

docker ps

Purpose

Lists containers.

Usage

```
docker ps
```

```
docker ps -a
```

docker ps shows running containers only.

docker ps -a includes stopped containers.

docker stop and docker start

Purpose

Stops or starts an existing container.

Usage

```
docker stop my-container
```

```
docker start my-container
```

docker rm

Purpose

Removes a stopped container.

Usage

```
docker rm my-container
```

docker inspect

Purpose

Displays low-level JSON metadata for containers or images.

Typical use cases

Inspect environment variables, network settings, mounts, ports, and runtime configuration.

Usage

```
docker inspect my-container
```

Executing Commands in Running Containers

docker exec

Purpose

Runs a command inside a running container.

Interactive shell

```
docker exec -it my-container bash
```

```
docker exec -it my-container sh
```

One-off command

```
docker exec my-container ls /etc
```

Logs and Runtime Monitoring

docker logs

Purpose

Displays standard output and error logs from a container.

Usage

```
docker logs my-container
```

```
docker logs -f my-container
```

```
docker logs -tail 50 my-container
```

docker stats

Purpose

Shows real-time CPU, memory, network, and disk usage for containers.

Usage

```
docker stats
```

```
docker stats my-container
```

Images and Registries

docker pull

Purpose

Downloads an image from a container registry.

Usage

```
docker pull ubuntu
```

```
docker pull nvcr.io/nvidia/pytorch:24.04-py3
```

docker images

Purpose

Lists all locally available images.

Usage

```
docker images
```

docker tag

Purpose

Creates an additional name and tag for an image, commonly used before pushing to a registry.

Usage

```
docker tag nginx myuser/nginx:demo
```

docker push

Purpose

Uploads an image to a container registry.

Usage

```
docker push myuser/nginx:demo
```

Prerequisite

You must authenticate with the registry using `docker login`.

Cleanup and Disk Management

docker system df

Purpose

Displays Docker disk usage information.

Usage

```
docker system df
```

docker container prune

Purpose

Removes all stopped containers.

Usage

```
docker container prune
```

docker image prune

Purpose

Removes dangling and unused images.

Usage

```
docker image prune
```

docker system prune

Purpose

Removes unused containers, networks, images, and build cache.

Usage

```
docker system prune
```

Aggressive cleanup

```
docker system prune -a --volumes
```

This removes all unused images and volumes and should be used with caution.

Docker Contexts

docker context ls

Purpose

Lists available Docker contexts such as local, desktop, or remote environments.

Usage

```
docker context ls
```

docker context use

Purpose

Switches the active Docker context.

Usage

```
docker context use desktop-linux
```

Docker Networking Basics

Docker Network Types

Default bridge network

Purpose

Provides a private, NATed network for containers. Containers attached to the same bridge network

can communicate using IP addresses. Name-based DNS resolution is limited on the default bridge.

Notes

This is the network used when no `-network` flag is specified.

Inspect the default bridge

```
docker network inspect bridge
```

Run a container on the default bridge

```
docker run -rm alpine sh
```

Host network

Purpose

Shares the host network stack with the container.

Notes

On macOS, Docker runs inside a virtual machine. As a result, `-network host` does not behave the same way it does on Linux and does not give direct access to the Mac's network namespace.

Run a container with host networking

```
docker run -rm -network host alpine sh
```

None network

Purpose

Disables all networking for the container except the loopback interface.

Run a container with no networking

```
docker run -rm -network none alpine sh
```

Managing Docker Networks

docker network ls

Purpose

Lists all Docker networks.

Usage

```
docker network ls
```

docker network inspect

Purpose

Displays detailed configuration and connected containers for a network.

Usage

`docker network inspect ai-net`

docker network create

Purpose

Creates a user-defined bridge network with built-in DNS resolution and isolation.

Usage

`docker network create ai-net`

Notes

User-defined bridge networks are recommended for multi-container applications.

docker network rm

Purpose

Deletes a Docker network.

Usage

`docker network rm ai-net`

Running Containers on Networks

Run container on a specific network

Purpose

Attaches a container to a specified network at startup.

Usage

`docker run -d --name web --network ai-net nginx`

docker network connect

Purpose

Connects an existing container to an additional network.

Usage

`docker network connect other-net web`

Notes

Containers can be attached to multiple networks simultaneously.

Container DNS Resolution

Purpose

Allows containers on the same user-defined bridge network to resolve each other by name using Docker's embedded DNS server.

Test DNS resolution

```
docker run -rm -it --network ai-net alpine sh -c "apk add --no-cache bind-tools && nslookup web"
```

Inspect DNS configuration inside a container

```
docker exec -it web cat /etc/resolv.conf
```

Add a network alias

```
docker network connect --alias webserv ai-net web
```

Port Publishing and Exposure

-p flag

Purpose

Publishes a container port to the host, enabling host-to-container communication.

Syntax

```
-p <host_port>:<container_port>
```

Example

```
docker run -d --name web -p 8080:80 nginx
```

Verify published ports

```
docker ps
```

```
docker port web
```

Notes

Port publishing is not required for container-to-container communication on the same Docker network.

Container-to-Container Communication

Same network communication

Purpose

Allows containers on the same network to communicate without published ports.

Test connectivity

```
docker run -rm -it --network ai-net nicolaka/netshoot sh -c "curl -I http://web"
```

Cross-network isolation

Purpose

Demonstrates that containers on different networks cannot communicate by default.

Test isolation

```
docker run -rm -it --network other-net nicolaka/netshoot sh -c "curl -I http://web || echo 'unreachable'"
```

Connect container to multiple networks

```
docker network connect other-net web
```

Re-test connectivity

```
docker run -rm -it -network other-net nicolaka/netshoot sh -c "curl -I http://web"
```

Networking Diagnostics and Debugging

Inspect container network settings

Purpose

Displays IP addresses, networks, and endpoints for a container.

Usage

```
docker inspect web
```

Use a diagnostic container

Purpose

Provides networking tools such as curl, ping, nslookup, and traceroute.

Usage

```
docker run -rm -it -network ai-net nicolaka/netshoot
```

Cleanup Networking Resources

Remove containers

```
docker rm -f web client
```

Remove networks

```
docker network rm ai-net test-net other-net
```