

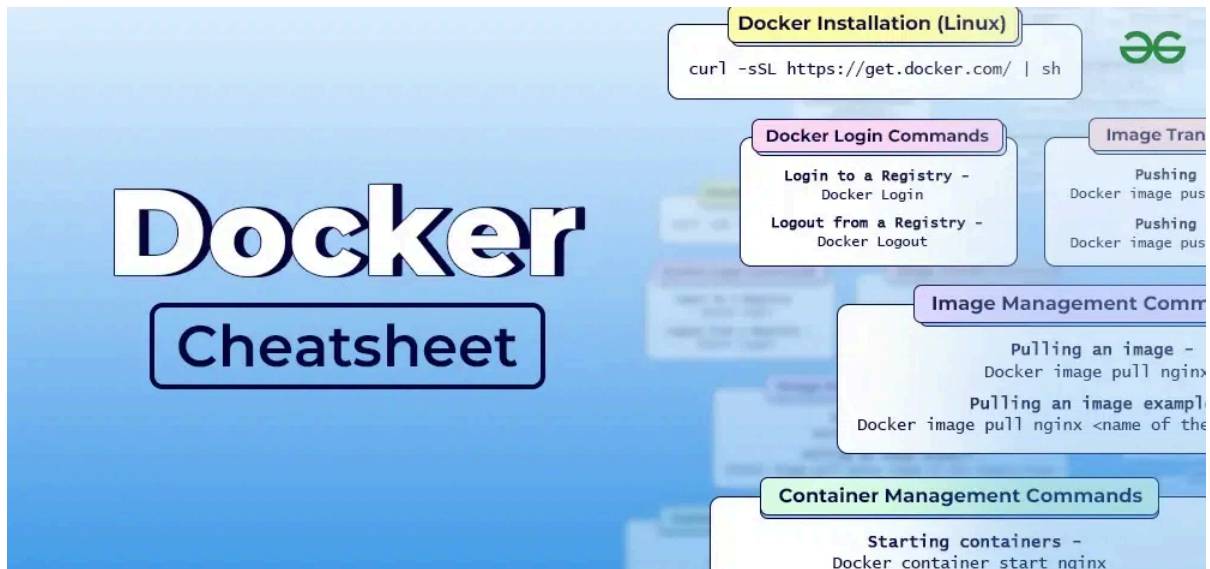


Docker Cheat Sheet : Complete Guide (2024)

Last Updated : 08 Oct, 2024

Docker is a very popular tool introduced to make it easier for developers to create, deploy, and run applications using containers. A container is a utility provided by Docker to package and run an application in a loosely isolated environment. Containers are lightweight and contain everything needed to run an application, such as libraries and other dependencies packed by the developer during the application's packaging process. This assures developers that their application can be run on any other machine. Here, we're going to provide you with an ultimate **Docker Cheat Sheet** that will help you to **learn Docker Commands** easily.

This **Docker command cheatsheet** is a summary of **commonly used Docker commands** and their options, as well as other useful information related to Docker. It covers all the important commands required for Docker operations, including Docker installation, building, running, shipping, and cleaning up, as well as interaction with containers. This cheat sheet will be a handy reference for you to perform various tasks with Docker, such as installing, building, running, shipping, and cleaning up containers. This **Docker cheat sheet** is useful for both – DevOps freshers who're learning Docker and experienced Docker users who need to recall a specific command or option but may not remember all the details.



What is Docker?

Docker allows you to collect and run an application in a container, which is a loosely isolated environment. Because of the isolation and security, you can run multiple containers on a single host at the same time. Containers are lightweight and include everything required to run the application, eliminating the need to rely on what is already installed on the host. You may easily share containers while working, and you can ensure that everyone with whom you share gets the same container that functions in the same way.

Pre-requisite: [Docker](#), [DockerHub](#)

The below Docker cheat sheet contains commands to manage the [docker containers](#), [images](#), [network](#), [volumes](#), building running, and deploying containers and also covered commands related to [Docker Compose](#).

Docker Cheat Sheet – Table of Content

Table of Content

- [Installation Commands](#)
- [Docker Login Commands](#)
- [Image Management Commands](#)
- [Image Transfer Commands](#)
- [Docker Hub Commands](#)

- [General Docker Commands](#)
- [Containers Management Commands](#)
- [Docker Image Management Commands](#)
- [Docker Network Commands](#)
- [Docker Exposing Ports Commands](#)
- [Docker Commands Removing Containers, Images, Volumes, And Networks](#)
- [Docker Swarm Commands](#)
- [Docker file Commands](#)
- [Docker Volume Commands](#)
- [Docker CP commands](#)
- [Docker Security Commands \(Docker Scout\)](#)

A **Docker cheat sheet** is a handy tool for quick reference, but mastering Docker requires hands-on practice. For an in-depth guide to using Docker in DevOps, the [DevOps Engineering – Planning to Production](#) course provides detailed instructions on Docker's full range of capabilities.

Docker Commands Cheat Sheet

The Docker cheat sheet will help you as a reference guide from where you can quickly read of mostly used common commands of Docker. The cheat sheet will help as a handy guide for developers and other system administrations who are working with Docker. Let's get started:

Installation Commands

Name	Command
Installation on Linux	<code>curl -sSL https://get.docker.com -o get-docker.sh && sudo sh get-docker.sh</code>

Docker Login Commands

Name	Command
Log in to a Registry	<code>docker login</code>
Logout from a Registry	<code>docker logout</code>

Image Management Commands

Docker images are self-contained software packages that contain all the necessary components to run an application. These components include the code, runtime, system tools, system libraries, and settings. Docker images are lightweight and easy to use.

Name	Command
Build an image	<code>docker build -t <image_name></code>

Name	Command
Pulling an Image	docker image pull nginx
Pulling an Image Example	docker image pull <Name of The Image>: <Tag>

Image Transfer Commands

Name	Command
Pushing an Image	docker image push <usernameofregistry:Imagename: tag>
Pushing an Image Example	docker image push eon01/nginx localhost:5000/myadmin/nginx

Docker Hub Commands

Docker Hub is a service provided by Docker for finding and sharing container images with your team. Learn more and find images at “<https://hub.docker.com>”.

Name	Command
Login into Docker	-docker login -u <username>
Publish an image to Docker Hub	-docker push <username>/<image_name>
Search Hub for an image	-docker search <image_name>

Name	Command
Pull an image from a Docker Hub	-docker pull <image_name>

General Docker Commands

Name	Command
Start the docker daemon	docker -d
Get help with Docker. Can also use <code>--help</code> on all subcommands	docker --help
Display system-wide information	docker info

Containers Management Commands

CONTAINERS

A docker image's runtime instance is referred to as a container. The container remains consistent regardless of the infrastructure in use. This isolation of software from its environment guarantees uniformity in function, even in cases where there are discrepancies between development and staging.

Name	Command
Starting Containers	docker container start nginx
Stopping Containers	docker container stop nginx

Name	Command
Restarting Containers	<code>docker container restart nginx</code>
Pausing Containers	<code>docker container pause nginx</code>
Unpausing Containers	<code>docker container unpause nginx</code>
Blocking a Container	<code>docker container wait nginx</code>
Sending SIGKILL Containers	<code>docker container kill nginx</code>
Sending another signal	<code>docker container kill -s HUP nginx</code>
Connecting to an Existing Container	<code>docker container attach nginx</code>
Check the Containers	<code>docker ps</code>
To see all running containers	<code>docker container ls</code>
Container Logs	<code>docker logs infinite</code>
'tail -f' Containers' Logs	<code>docker container logs infinite -f</code>
Inspecting Containers	<code>docker container inspect infinite</code>
Inspecting Containers for certain	<code>docker container inspect --format '{{.NetworkSettings.IPAddress}}' \$(docker ps -q)</code>

Name	Command
Containers Events	<code>docker system events infinite</code>
<code>docker system events infinite</code>	<code>docker container port infinite</code>
Running Processes	<code>docker container top infinite</code>
Container Resource Usage	<code>docker container stats infinite</code>
Inspecting changes to files or directories on a container's filesystem	<code>docker container diff infinite</code>

Docker Image Management Commands

Name	Command
Listing Images	<code>docker image ls</code>
Building Images	<code>docker build.</code>
From a Remote GIT Repository	<code>docker build github.com/creack/docker-firefox</code>
Instead of Specifying a Context, You Can Pass a Single Dockerfile in the URL or Pipe the File in via STDIN	<code>docker build - < Dockerfile</code>

Name	Command
Building and Tagging	<code>docker build -t eon/infinite.</code>
Building a Dockerfile while Specifying the Build Context	<code>docker build -f myOtherDockerfile.</code>
Building from a Remote Dockerfile URI	<code>curl example.com/remote/Dockerfile docker build -f - .</code>
Removing an Image	<code>docker image rm nginx</code>
Loading a Tarred Repository from a File or the Standard Input Stream	<code>docker image load < ubuntu.tar.gz</code>
Saving an Image to a Tar Archive	<code>docker image save busybox > ubuntu.tar</code>
Showing the History of an Image	<code>docker image history</code>
Creating an Image From a Container	<code>docker container commit nginx</code>
Tagging an Image	<code>docker image tag nginx eon01/nginx</code>
Pushing an Image	<code>docker image push eon01/nginx</code>

Docker Network Commands

Name	Command
Creating an Overlay Network	<code>docker network create -d overlay MyOverlayNetwork</code>
Creating a Bridge Network	<code>docker network create -d bridge MyBridgeNetwork</code>
Creating a Customized Overlay Network	<code>docker network create -d overlay \</code> <code> --subnet=192.168.0.0/16 \</code> <code> --subnet=192.170.0.0/16 \</code> <code> --gateway=192.168.0.100 \</code> <code> --gateway=192.170.0.100 \</code> <code> --ip-range=192.168.1.0/24 \</code> <code> --aux-address="my-router=192.168.1.5"</code> <code> --aux-address="my-switch=192.168.1.6" \</code> <code> --aux-address="my-printer=192.170.1.5"</code> <code> --aux-address="my-nas=192.170.1.6" \</code> <code> MyOverlayNetwork</code>
Removing a Network	<code>docker network rm MyOverlayNetwork</code>
Listing Networks	<code>docker network ls</code>
Getting Information About a Network	<code>docker network inspect MyOverlayNetwork</code>
Connecting a Running Container to a Network	<code>docker network connect MyOverlayNetwork nginx</code>

Name	Command
Connecting a Container to a Network When it Starts	<code>docker container run -it -d --network=MyOverlayNetwork nginx</code>
Disconnecting a Container from a Network	<code>docker network disconnect MyOverlayNetwork nginx</code>

Docker Exposing Ports Commands

Name	Command
Exposing Ports	<code>EXPOSE <port_number></code>
Mapping Ports	<code>docker run -p \$HOST_PORT:\$CONTAINER_PORT --name <container_name> -t <image></code>

Docker Commands Removing Containers, Images, Volumes, And Networks

Name	Command
Removing a Running Container	<code>docker container rm nginx</code>
Removing a Container and its Volume	<code>docker container rm -v nginx</code>
Removing all Exited Containers	<code>docker container rm \$(docker container ls -a -f status=exited -q)</code>

Name	Command
Removing All Stopped Containers	<code>docker container rm `docker container ls -a -q`</code>
Removing a Docker Image	<code>docker image rm nginx</code>
Removing Dangling Images	<code>docker image rm \$(docker image ls -f dangling=true -q)</code>
Removing all Images	<code>docker image rm \$(docker image ls -a -q)</code>
Removing all Untagged Images	<code>docker image rm -f \$(docker image ls grep "^" awk "{print \$3}")</code>
Stopping & Removing all Containers	<code>docker container stop \$(docker container ls -a -q) && docker container rm \$(docker container ls -a -q)</code>
Removing Dangling Volumes	<code>docker volume rm \$(docker volume ls -f dangling=true -q)</code>
Removing all unused (containers, images, networks and volumes)	<code>docker system prune -f</code>
Clean all	<code>docker system prune -a</code>

Docker Swarm Commands

Name	Command
Installing Docker Swarm	<code>curl -ssl https://get.docker.com bash</code>
Initializing the Swarm	<code>docker swarm init --advertise-addr 192.168.10.1</code>
Getting a Worker to Join the Swarm	<code>docker swarm join-token worker</code>
Getting a Manager to Join the Swarm	<code>docker swarm join-token manager</code>
Listing Services	<code>docker service ls</code>
Listing nodes	<code>docker node ls</code>
Creating a Service	<code>docker service create --name vote -p 8080:80 instavote/vote</code>
Listing Swarm Tasks	<code>docker service ps</code>
Scaling a Service	<code>docker service scale vote=3</code>
Updating a Service	<code>docker service update --image instavote/vote:movies vote</code>
Updating a Service	<code>docker service update --force --update-parallelism 1 --update-delay 30s nginx</code>

Docker file Commands

Command	Description	Example
FROM	Specifies the base image for the build	FROM ubuntu:latest
RUN	Executes a command inside the container during build time	RUN apt-get update && apt-get install -y curl
CMD	Specifies the default command to run when the container starts	CMD ["npm", "start"]
EXPOSE	Informs Docker that the container listens on specific network ports at runtime	EXPOSE 80/tcp
ENV	Sets environment variables inside the container	ENV NODE_ENV=production
COPY	Copies files or directories from the build context into the container	COPY app.js /usr/src/app/
ADD	Similar to COPY but supports additional features like URL retrieval and decompression	ADD https://example.com/file.tar.gz /usr/src/

Command	Description	Example
WORKDIR	Sets the working directory for subsequent instructions	WORKDIR /usr/src/app
ARG	Defines variables that users can pass at build-time to the builder with the docker build command	ARG VERSION=1.0
ENTRYPOINT	Configures a container to run as an executable	ENTRYPOINT ["python", "app.py"]
VOLUME	Creates a mount point and assigns it to a specified volume	VOLUME /data
USER	Sets the user or UID to use when running the image	USER appuser
LABEL	Adds metadata to an image in the form of key-value pairs	LABEL version="1.0" maintainer="John Doe"

Command	Description	Example
ONBUILD	Configures commands to run when the image is used as the base for another build	ONBUILD ADD ./app/src

Docker Volume Commands

Command	Description	Example
volume create	Creates a named volume	docker volume create mydata
volume ls	Lists the available volumes	docker volume ls
volume inspect	Displays detailed information about a volume	docker volume inspect mydata
volume rm	Removes one or more volumes	docker volume rm mydata
volume prune	Removes all unused volumes	docker volume prune

Docker CP commands

Command	Description	Example
docker cp [OPTIONS] SRC_PATH CONTAINER:DEST_PATH	Copies files or directories from the local filesystem to the specified container	<code>docker cp myf mycontainer:/usr/src/app/</code>
docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH	Copies files or directories from the specified container to the local filesystem	<code>docker mycontainer:/usr/src/app/res /tmp/result/</code>

Docker Security Commands (Docker Scout)

Command	Description	Example
docker scout compare	[experimental] Compare two images and display differences	<code>docker scout compare image1:tag image2:tag</code>
docker scout cves	Display CVEs identified in a software artifact	<code>docker scout cves image: tag</code>
docker scout Quickview	Quick overview of an image	<code>docker scout quickview image: tag</code>

Command	Description	Example
<code>docker scout recommendations</code>	Display available base image updates and remediation recommendations	<code>docker scout recommendations image:tag</code>
<code>docker scout version</code>	Show Docker Scout version information	<code>docker scout version</code>

Conclusion

In conclusion, this **Docker cheat sheet** helps you with a **quick revision of all the Docker commands** that are required for Docker operations, including Docker installation, building, running, shipping, and cleaning up, as well as interaction with containers.

Docker CheatSheet – FAQs

1. What is the architecture of Docker?

Answer:

Docker follows a client-server architecture. The Docker client communicates with the Docker daemon, which is responsible for building, running, and managing Docker containers. The client and daemon can run on the same host, or the client can connect to a remote daemon.

2. Which language is Docker built on?

Answer:

Docker is built using Go programming language because of its advantage of several features of the Linux kernel to deliver its functionality.

3. Does Docker require coding?

Answer:

No, Docker does not require any prior coding knowledge. It is a containerization platform that enables developers to package, deploy, and run applications using containers.

4. Are Docker secrets safe?

Answer:

You can use Docker secrets to centrally manage this data and securely transmit it to only those containers that need access to

it. Secrets are encrypted during transit and at rest in a Docker swarm.

5. How many types of volumes are there in Docker?

Answer:

Docker supports three types of volumes:

*a) **Named Volumes:** These are volumes with a user-defined name that can be used across multiple containers.*

*b) **Bind Mounts:** These are directories on the host machine that are mounted into a container, allowing direct access to the host's file system.*

*c) **tmpfs Mounts:** These are volumes stored in the host's memory, allowing fast read and write operations but with limited size and durability.*

6. What is the flag in Docker?

Answer:

In Docker, a flag is a command-line option that modifies the behavior of a Docker command. Flags are used to provide additional instructions or parameters to Docker commands, allowing you to customize the execution according to your needs.

7. Why is Docker used in DevOps?

Answer:

Docker is widely used in DevOps practices due to its ability to create reproducible and portable environments. With Docker,

developers can package their applications and dependencies into containers, ensuring consistent behavior across different stages of the software development lifecycle. Docker also facilitates the automation of deployment, testing, and scaling processes, enabling faster and more reliable software delivery in DevOps pipelines.



GeeksforGeeks



6

Next Article

Architecture of Docker

Similar Reads

jQuery Cheat Sheet – A Basic Guide to jQuery

What is jQuery?jQuery is an open-source, feature-rich JavaScript library, designed to simplify the HTML document traversal and manipulation,...

15+ min read

CSS Cheat Sheet - A Basic Guide to CSS

What is CSS? CSS i.e. Cascading Style Sheets is a stylesheet language used to describe the presentation of a document written in a markup...

13 min read

Angular Cheat Sheet - A Basic Guide to Angular

Angular is a client-side TypeScript-based, front-end web framework developed by the Angular Team at Google, that is mainly used to...

15+ min read

Bootstrap Cheat Sheet - A Basic Guide to Bootstrap

Bootstrap is a free, open-source, potent CSS framework and toolkit used to create modern and responsive websites and web applications. It is t...

15+ min read

JavaScript Cheat Sheet - A Basic Guide to JavaScript

JavaScript is a lightweight, open, and cross-platform programming language. It is omnipresent in modern development and is used by...

15+ min read

Python Cheat sheet (2024)

Python is one of the most widely-used and popular programming languages, was developed by Guido van Rossum and released first in...

15+ min read

Tkinter Cheat Sheet

Tkinter, the standard GUI library for Python, empowers developers to effortlessly create visually appealing and interactive desktop...

8 min read

ggplot2 Cheat Sheet

Welcome to the ultimate ggplot2 cheat sheet! This is your go-to resource for mastering R's powerful visualization package. With ggplot...

13 min read

C Cheat Sheet

This C Cheat Sheet provides an overview of both basic and advanced concepts of the C language. Whether you're a beginner or an...

15+ min read

NumPy Cheat Sheet: Beginner to Advanced (PDF)

NumPy stands for Numerical Python. It is one of the most important foundational packages for numerical computing & data analysis in...

15+ min read

Article Tags :

[DevOps](#)[Docker](#)[docker](#)[GFG Sheets](#)



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation

[Top 100 DSA Interview Problems](#)[DSA Roadmap by Sandeep Jain](#)[All Cheat Sheets](#)[Pandas](#)[NumPy](#)[NLP](#)[Deep Learning](#)

Web Technologies

[HTML](#)[CSS](#)[JavaScript](#)[TypeScript](#)[ReactJS](#)[NextJS](#)[Bootstrap](#)[Web Design](#)

Python Tutorial

[Python Programming Examples](#)[Python Projects](#)[Python Tkinter](#)[Web Scraping](#)[OpenCV Tutorial](#)[Python Interview Question](#)[Django](#)

Computer Science

[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)[Software Development](#)[Software Testing](#)

DevOps

[Git](#)[Linux](#)[AWS](#)[Docker](#)[Kubernetes](#)[Azure](#)[GCP](#)[DevOps Roadmap](#)

System Design

[High Level Design](#)[Low Level Design](#)[UML Diagrams](#)[Interview Guide](#)[Design Patterns](#)[OOAD](#)[System Design Bootcamp](#)[Interview Questions](#)

Interview Preparation

[Competitive Programming](#)[Top DS or Algo for CP](#)[Company-Wise Recruitment Process](#)[Company-Wise Preparation](#)[Aptitude Preparation](#)[Puzzles](#)

School Subjects

[Mathematics](#)[Physics](#)[Chemistry](#)[Biology](#)[Social Science](#)[English Grammar](#)[Commerce](#)[World GK](#)

GeeksforGeeks Videos

[DSA](#)[Python](#)[Java](#)[C++](#)[Web Development](#)[Data Science](#)[CS Subjects](#)

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved