perplexity

Docker Command Reference: Expanded Cheat Sheet & Workflow Guide

Docker's CLI now spans hundreds of sub-commands, plugins, and orchestration features. This extended reference groups the most- and often-forgotten commands into logical workflows—helping beginners ramp up quickly while giving seasoned engineers a single, searchable source for day-to-day operations.

Overview

Docker simplifies packaging, shipping, and running applications by wrapping them inside isolated containers. Much of that power hides behind a versatile command-line interface. This guide expands the classic cheat sheet with deeper coverage of:

- Installation diagnostics, context switching, and daemon inspection.
- Fine-grained image, container, volume, network, and build cache management.
- Modern BuildKit/Buildx workflows, multi-platform publishing, and inline cache exports.
- Compose v2 sub-commands, environment overrides, and configuration helpers.
- System pruning, disk-usage analytics, and resource monitoring.
- On-the-fly debugging, logs, file copy, and interactive maintenance tasks.
- Swarm orchestration starters and context-aware remote management.

Installation & Initial Diagnostics

Command	Purpose
`docker --version`	Display Docker Engine client/server versions [1].
`docker info`	Full daemon diagnostics: storage driver, kernel, cgroup version, container/image counts [2].
`docker context ls`	List contexts (local, remote SSH, cloud, Kubernetes) [3].
`docker context use <name>`	Switch the active context for subsequent commands [4].
`docker compose version`	Show Compose CLI plugin build & API compatibility [5].
`docker system info`	Summarize resources, driver plugins, registries, default root dir [2].

Quick Health Check

Run:

```
docker info --format '{{.ServerVersion}} — {{.OperatingSystem}}'
```

to ensure the daemon is reachable and exporting metrics.

Core Docker Concepts

- **Image**: A layered, read-only template created from a **Dockerfile**. [6]
- **Container**: An isolated runtime instance of an image. [7]
- **Volume**: Managed data storage independent of the container's writable layer. [8]
- **Network**: Virtual switch enabling secure container-to-container communication. [9]
- **Context**: Named set of endpoint credentials for talking to multiple daemons, swarms, or K8s clusters from one CLI. [3]
- **BuildKit/Buildx**: Next-gen builder backend & CLI plugin enabling parallel, multi-arch builds with advanced caching. [10]
- **Compose**: YAML-driven orchestration tool for multi-container stacks. [5]
- **Swarm**: Native clustering/orchestration mode built into Docker Engine. [11]

General CLI Help & Context Management

Command	Description
`docker help`	Top-level command index with global flags [2].
`docker <subcmd> --help`	Detailed usage, options, and examples for any sub-command [2].
`docker context create <name> --docker host=ssh://user@host`	Register a remote daemon over SSH [4].
`docker context inspect <name>`	Show endpoints, TLS data, storage paths [3].
`docker context show`	Print the name of the current context (handy for prompts) [12].
`docker context export <name> ctx.tar`	Archive a context for sharing/dr [3].

Image Lifecycle Commands

Command	Function
`docker build -t app:1.0 .`	Build image in current dir with tag [6].
`docker build --platform linux/arm64 -t app:arm .`	Cross-compile with BuildKit/Buildx [10].
`docker image ls`	List images; `-a` shows intermediate layers [13].

Command	Function
`docker image history <id>`	Display layer ancestry & commands [6].
`docker pull alpine:3.20`	Retrieve from registry [6].
`docker tag <img> user/repo:prod`	Retag for push [6].
`docker push user/repo:prod`	Upload to registry [14].
`docker image inspect <id>`	Manifest & config JSON [6].
`docker image rm <id>`	Delete unused/cached images [6].
`docker image prune -a --filter until=168h`	Purge dangling & aged images [15].
`docker save app:1.0 -o app.tar`	Export image to tar archive [16].
`docker load -i app.tar`	Import image archive [16].

Container Runtime Management

Command	Purpose
`docker run -d -p 8080:80 --name web nginx:stable`	Start container detached with port mapping [7].
`docker exec -it web bash`	Open interactive shell in running container [17].
`docker container ls`	List active containers (alias: `docker ps`) [7].
`docker container ls -a`	Show all including stopped [7].
`docker logs -f --tail 100 web`	Stream logs with follow [7].
`docker top web`	Show container processes [7].
`docker stats web`	Live CPU/MEM/IO metrics [18].
`docker stop web`	Graceful SIGTERM then SIGKILL after timeout [7].
`docker restart web`	Convenience stop → start sequence [7].
`docker container inspect web`	Low-level config/state (Mounts, PID, IPs) [7].
`docker rename web api-web`	Change container name [7].
`docker commit web debug:tmp`	Snapshot changes into new image [7].
`docker rm web`	Remove stopped container [7].
`docker cp web:/var/log/nginx access.log`	Copy file from container to host [19].
`docker cp config.yml web:/app/`	Push file into running container [20].

Volume Commands

Command	Role
`docker volume create pgdata`	Provision named volume [21].
`docker volume ls`	Enumerate volumes [21].
`docker volume inspect pgdata`	Path, driver, mount-point [21].
`docker volume rm pgdata`	Delete volume (must be unused) [21].
`docker volume prune --force`	Remove all dangling volumes [21].
`docker run -v pgdata:/var/lib/postgresql/data postgres:16`	Mount volume into container [8].

Network Management

Command	Action
`docker network ls`	List user & default networks [9].
`docker network create --driver bridge app-net`	Create bridge network [9].
`docker network inspect app-net`	CIDR, containers, gateways [9].
`docker network connect app-net web`	Attach running container [9].
`docker network disconnect bridge debug-box`	Detach container [9].
`docker network prune`	Delete unused networks [9].

BuildKit & Buildx Workflows

Build Command	Highlights
`docker buildx create --name multi --use`	Spin up separate builder instance [22].
`docker buildx build --platform linux/amd64,linux/arm64 -t user/app:latest --push .`	Multi-arch build & direct push [22].
`docker buildx bake -f docker-bake.hcl release`	Declarative batch builds via bake file [23].
`docker buildx du`	Estimate build cache size [22].
`docker buildx prune --filter until=72h --force`	Remove aged cache layers [23].
`docker builder prune --all --keep-storage 20GB`	Legacy builder cache cleanup [24].

> **Tip:** Set `DOCKER_BUILDKIT=1` in your shell profile so legacy `docker build` automatically leverages BuildKit improvements.

Dockerfile Instruction Quick-Ref

Directive	Purpose
`FROM`	Base image; first instruction required [25].
`RUN`	Execute shell commands during build [25].
`COPY`	Add local files/directories to image [25].
`ADD`	Like COPY + remote URLs & archives [25].
`CMD`	Default container command (overridden by run args) [25].
`ENTRYPOINT`	Preferred executable, arguments appended [25].
`ARG`	Build-time variable (not persisted) [25].
`ENV`	Environment variable available at runtime [25].
`EXPOSE`	Documented port(s) the app listens on [25].
`WORKDIR`	Set working directory [25].
`VOLUME`	Declare mount point for volumes [25].
`HEALTHCHECK`	Periodic check returning healthy/unhealthy [25].
`USER`	Switch UID/GID for subsequent layers [25].

Compose v2 Sub-Commands

Command	Function
`docker compose up`	Build (if needed) and start all services [5].
`docker compose up -d --scale worker=3`	Detached start with replica override [26].
`docker compose logs -f --tail=50`	Live logs across services [5].
`docker compose ps`	Container status & port mappings [5].
`docker compose down -v --rmi all`	Stop stack, remove volumes & images [5].
`docker compose restart api`	Recycle single service [5].
`docker compose exec db psql -U user appdb`	Exec into service container [5].
`docker compose config --profiles dev`	Render merged YAML for selected profile [5].
`docker compose cp web:/usr/share/nginx/html ./dist`	Copy from service container [5].
`docker compose watch`	Rebuild/restart on file changes (dev loop) [5].

Monitoring & Debugging

Command	Insight
`docker stats --no-stream`	One-shot resource snapshot for all containers[27].
`docker stats --format "table {{.Name}}\t{{.CPUPerc}}\t{{.MemUsage}}"`	Custom column output[18].
`docker logs --since 10m api`	Recent logs only[7].
`docker events --filter container=web`	Real-time daemon event stream[7].
`docker container inspect --format '{{.State.ExitCode}}' failing`	Extract JSON fields quickly[7].
`docker debug <container>`	Drop into ephemeral toolbox shell even in slim images[28].

Cleanup & Disk-Usage

Command	What it Removes
`docker system df`	Show disk footprint per object type (safe to run)[2].
`docker system prune`	Stopped containers, dangling images, unused networks & build cache[15].
`docker system prune -a --volumes`	EVERYTHING unused, including volumes (use with care)[29].
`docker container prune --filter "until=24h"`	Only containers older than a day[30].
`docker image prune --filter "label!=stage=builder"`	Keep builder layers by label[31].
`docker builder prune --keep-storage 5GB`	Maintain 5GB cache while purging rest[24].

Swarm Quick-Start

Command	Task
`docker swarm init --advertise-addr <MANAGER_IP>`	Bootstrap a new swarm[32].
`docker swarm join --token <TOKEN> <MANAGER_IP>:2377`	Add worker/manager nodes[33].
`docker node ls`	List nodes and roles[32].
`docker service create --name web --replicas 3 -p 80:80 nginx`	Deploy replicated service[34].
`docker service ps web`	Check task allocation & failures[34].
`docker service scale web=5`	Dynamically change replica count[34].
`docker stack deploy -c docker-compose.yml mystack`	Deploy multi-service stack[34].

Command	Task
`docker swarm leave --force`	Remove current node from swarm [11].

Best-Practice Tips

- **Keep Dockerfiles cache-friendly:** Copy only dependency manifests (e.g., `package.json`, `requirements.txt`) before full source to leverage layer reuse. [25]
- **Label intermediate builder stages:** `LABEL stage=builder` then filter prunes to preserve heavy build layers. [31]
- **Use named volumes instead of anonymous:** Easy backup & avoids unintentional deletion. [8]
- **Pin base image tags:** Relying on `latest` risks surprise rebuilds and security drift. [14]
- **Scan images for CVEs:** Pair with `docker scan` or third-party scanners in CI (not covered in this sheet).
- **Automate cleanup:** Schedule `docker system prune -f --filter "until=168h"` weekly on dev machines. [15]
- **Context-aware prompts:** Show current `docker context` in your shell to avoid pushing images to the wrong environment. [12]

Glossary of Common Flags

Flag	Applies To	Meaning
`-d, --detach`	`run`, `exec`, Compose	Run in background [7].
`-p host:cont`	`run`, Compose	Map TCP/UDP port [7].
`--rm`	`run`	Auto-remove container on exit [7].
`--no-cache`	`build`	Ignore layer cache [6].
`--platform`	`build`, `run`	Target CPU/OS architecture [10].
`--filter`	`ps`, `prune`, `events`	Conditional output/prune [15].
`--scale svc=N`	Compose	Override replica count at runtime [26].
`--env KEY=VAL`	`run`, Compose	Set env variable [7].
`--mount type=bind,src=...,dst=...`	`run`	Explicit bind mount syntax [8].

Putting It All Together — Example Workflow

1. **Initialize project**

```
git clone https://github.com/user/app && cd app
docker buildx create --name multi --use
docker buildx build --platform linux/amd64,linux/arm64 \
    -t user/app:1.0 --push .
```

2. **Launch local stack**

```
docker compose up -d
docker stats --no-stream
```

3. **Iterate on code**
   - Edit sources, then rely on `docker compose watch` for hot-reload. [5]
   - Attach with `docker debug api` for troubleshooting.[28]
4. **Ship to staging**

```
docker context use staging
docker stack deploy -c deploy.yml app
```

5. **Routine maintenance**

```
docker context use default
docker system df
docker system prune --filter "until=168h" -f
docker buildx prune --filter until=168h -f
```

Closing Thoughts

Anchored in the official Docker documentation and community best practices, this expanded cheat sheet should serve as both a quick reference and a deep-dive tutorial. Keep experimenting—Docker's CLI evolves rapidly with BuildKit, Compose v2, and new orchestration primitives unlocking fresh possibilities every release.

Continuous learning and thoughtful cleanup go hand-in-hand with containerized development. Bookmark this guide, share it with teams, and revisit after each Docker upgrade to stay sharp and efficient.

⁂

1. https://www.docker.com/products/cli/
2. https://spacelift.io/blog/docker-commands-cheat-sheet
3. https://docs.docker.com/engine/manage-resources/contexts/
4. https://docs.docker.com/reference/cli/docker/context/
5. https://docs.docker.com/reference/cli/docker/compose/
6. https://docs.docker.com/reference/cli/docker/image/
7. https://docs.docker.com/reference/cli/docker/container/
8. https://docs.docker.com/engine/storage/volumes/
9. https://docs.docker.com/reference/cli/docker/network/
10. https://docs.docker.com/build/concepts/overview/
11. https://docs.docker.com/reference/cli/docker/swarm/
12. https://docs.docker.com/reference/cli/docker/context/show/

13. https://docs.docker.com/reference/cli/docker/image/ls/
14. https://docs.docker.com/get-started/docker-concepts/building-images/build-tag-and-publish-an-image/
15. https://docs.docker.com/reference/cli/docker/system/prune/
16. https://www.geeksforgeeks.org/devops/using-cli-to-manage-docker-volumes/
17. https://docs.docker.com/reference/cli/docker/container/exec/
18. https://docs.docker.com/reference/cli/docker/container/stats/
19. https://docs.docker.com/reference/cli/docker/container/cp/
20. https://kodekloud.com/blog/docker-cp/
21. https://docs.docker.com/reference/cli/docker/volume/
22. https://docs.docker.com/reference/cli/docker/buildx/
23. https://docs.docker.com/reference/cli/docker/buildx/prune/
24. https://docs.docker.com/reference/cli/docker/builder/prune/
25. https://docs.docker.com/reference/dockerfile/
26. https://gist.github.com/mkfares/41c9609fcde8d9f665210034e99d4bd9
27. https://docker-docs.uclv.cu/engine/reference/commandline/stats/
28. https://docs.docker.com/reference/cli/docker/debug/
29. https://depot.dev/blog/docker-system-prune
30. https://www.geeksforgeeks.org/devops/docker-prune/
31. https://www.reddit.com/r/docker/comments/12zojq6/prune_all_unused_images_and_containers_but_not/
32. https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/
33. https://learn.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/swarm-mode
34. https://doc.nexusgroup.com/pub/docker-swarm-commands