

# Memorize these 7 commands to stop messing around with Docker containers.

But using Docker effectively isn't just about installing and running containers. You'll get the most out of Docker when you know the commands that give you visibility, control, and efficiency.

Docker allows applications and their dependencies to run reliably on a computer by packaging them into containers, which are portable, isolated environments. This is an ideal solution if you want a reliable, repeatable workflow without worrying about whether a tool or system will work on your machine.

But using Docker effectively isn't just about installing and running containers. You'll get the most out of Docker when you know the commands that give you visibility, control, and efficiency.

## Docker Compose

Run multi-service environments with confidence.

```

fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo
├─ Network docker-demo default
├─ Container docker-demo-api-1
└─ Container docker-demo-web-1
Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint docker-demo-web-1 (e01c5599c31a4ab0991b178ef47f3f5ecb5b9ad5a6c1f1dd94e35fb66101d3): Bind for 0.0.0.0:8080 failed: port is already allocated
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ sudo lsuf -l :8080
[sudo] password for fuzo:
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
docker-pr 1981 root 7u IPv4 11033 0t0 TCP *:http-alt (LISTEN)
docker-pr 2002 root 7u IPv6 11034 0t0 TCP *:http-alt (LISTEN)
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ff776cff1b9 alpine "sh -c 'while true; _" About a minute ago Up About a minute docker-demo
-api-1
07279a7c1933 nextcloud:29 "/entrypoint.sh apac..." 3 months ago Up 19 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp nextcloud-l
ocal-app-1
be3740d5c04 mariadb:11 "docker-entrypoint.s..." 3 months ago Up 19 minutes 3306/tcp nextcloud-l
ocal-db-1
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker compose up -d
├─ Container docker-demo-web-1 Started
└─ Container docker-demo-api-1 Running
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d17b6102c67f nginx "/docker-entrypoint..." About a minute ago Up About a minute 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp docker-demo
-web-1
ff776cff1b9 alpine "sh -c 'while true; _" 4 minutes ago Up 4 minutes docker-demo
-api-1
07279a7c1933 nextcloud:29 "/entrypoint.sh apac..." 3 months ago Up 22 minutes 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp nextcloud-l
ocal-app-1
be3740d5c04 mariadb:11 "docker-entrypoint.s..." 3 months ago Up 22 minutes 3306/tcp nextcloud-l
ocal-db-1
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$

```

Before using the Compose command, people typically ran each container individually and could only hope they connected correctly. This approach was manual and often led to a buggy workflow. However, using the Compose command transformed Docker into an automated workflow, with just a single command, where services, images, ports, environment variables, and volumes are defined in the docker-compose.yml file.

The following command allows you to start the entire stack in the background:

```
docker compose up -d
```

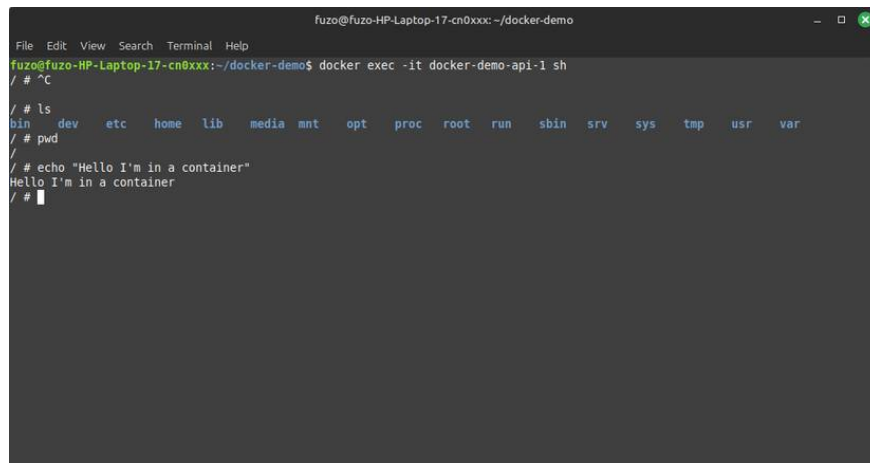
After testing, you can turn everything off using the command:

```
docker compose down
```

With these two simple commands, you'll have better control over Docker.

## Docker exec -it bash

### Troubleshooting containers from the inside

A terminal window titled 'fuza@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo' showing the execution of 'docker exec -it docker-demo-api-1 sh'. The prompt changes to '# ^C' and then to a shell prompt '#'. The user enters 'ls' and the output is 'bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var'. Then the user enters 'pwd' and the output is '/'. Finally, the user enters 'echo "Hello I'm in a container"' and the output is 'Hello I'm in a container'. The prompt returns to '#'.

```
fuza@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo
File Edit View Search Terminal Help
fuza@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker exec -it docker-demo-api-1 sh
/# ^C
/#
/# ls
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
/# pwd
/#
/# echo "Hello I'm in a container"
Hello I'm in a container
/#
```

It's essential to know how to troubleshoot Docker containers, and here's a commonly used command. Use it when you need to check file paths, verify configurations, run quick tests, or debug problems right where they occur:

```
docker exec -it myapp bash
```

If bash is not available, you can switch to this command:

```
docker exec -it myapp sh
```

Using either of these commands will open an interactive shell, allowing you to explore the container's actual environment.

## Docker logs -f

**Read the journal directly and stop guessing.**

```
fuze@fuze-HP-Laptop-17-cn0xxx:~$ docker logs -f docker-demo-web-1
hello
hello
hello
hello
hello
^Cfuze@fuze-HP-Laptop-17-cn0xxx:~$ docker logs -f docker-demo-api-1
docker logs -f docker-demo-api-1
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/12/12 11:26:38 [notice] #1: using the "epoll" event method
2025/12/12 11:26:38 [notice] #1: nginx/1.29.4
2025/12/12 11:26:38 [notice] #1: built by gcc 14.2.0 (Debian 14.2.0-19)
2025/12/12 11:26:38 [notice] #1: OS: Linux 6.8.0-87-generic
2025/12/12 11:26:38 [notice] #1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/12/12 11:26:38 [notice] #1: start worker processes
2025/12/12 11:26:38 [notice] #1: start worker process 29
2025/12/12 11:26:38 [notice] #1: start worker process 30
2025/12/12 11:26:38 [notice] #1: start worker process 31
2025/12/12 11:26:38 [notice] #1: start worker process 32
2025/12/12 11:26:38 [notice] #1: start worker process 33
2025/12/12 11:26:38 [notice] #1: start worker process 34
2025/12/12 11:26:38 [notice] #1: start worker process 35
2025/12/12 11:26:38 [notice] #1: start worker process 36
```

Logging provides a useful way to detect potential problems, and this Docker command is very handy:

```
docker logs -f myapp
```

With this command, you can access logs when a container fails to start. Add the `-f` flag to keep the logs streamed live. This allows you to track the container's startup sequence. This makes it easier to spot missing database login credentials, misspelled environment variables, or poorly configured ports.

Sometimes, logs can be long, making it difficult to find specific elements or troubleshoot problems. Therefore, limit the log results you see to the last 50 lines using the command below:

```
docker logs --tail 50 myapp
```

## Docker build

### Build predictable images with proper tagging.

```
fuze@fuze-HP-Laptop-17-cn0xxx:~/docker-demo$ docker build -t myapp .
docker: command not found
fuze@fuze-HP-Laptop-17-cn0xxx:~$ docker build -t myapp:v1 .
[+] Building 2.3s (1/1) FINISHED
    => [internal] load build definition from Dockerfile
    => [internal] load build definition from Dockerfile 0.0s
    => [internal] load build definition from Dockerfile 0.0s
ERROR: failed to build: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
fuze@fuze-HP-Laptop-17-cn0xxx:~$ cd ~/docker-demo
Command 'cd' not found, did you mean:
  command 'bcd' from deb hfsutils (3.2.6-15build2)
  command 'bcd' from deb bsdgames (2.17-30)
  command 'mcd' from deb mtools (4.0.43-1)
Try: sudo apt install <deb name>
fuze@fuze-HP-Laptop-17-cn0xxx:~$ cd ~/docker-demo
fuze@fuze-HP-Laptop-17-cn0xxx:~/docker-demo$ ls
docker-compose.yml
fuze@fuze-HP-Laptop-17-cn0xxx:~/docker-demo$ docker build -t myapp .
[+] Building 1.7s (1/1) FINISHED
    => [internal] load build definition from Dockerfile
    => [internal] load build definition from Dockerfile 0.0s
    => [internal] load build definition from Dockerfile 0.0s
```

The `build` command instructs the Docker daemon to begin the image creation process. When first starting with Docker, builds are often messy. People often don't tag anything, and quickly end up with many unlabeled images. The command below will fix your messy, untagged builds:

```
docker build -t myapp .
```

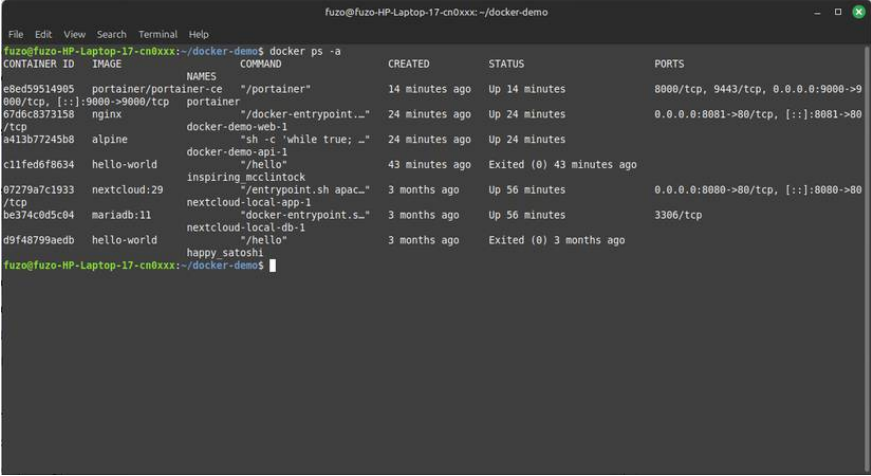
Adding the **-t** flag will tag the resulting image, making it easier to reuse and deploy. Taking it a step further, you can add more versions using the command below:

```
docker build -t myapp:v1 .
```

With this, I can test features without losing track of stable builds. Another benefit of tagging is preventing clutter, eliminating "floating" images that waste space.

## Docker ps -a

### Stop losing track of containers.



```
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
e8ed59514905   portainer/portainer-ce             "/portainer"           14 minutes ago Up 14 minutes 8000/tcp, 9443/tcp, 0.0.0.0:9000->9
000/tcp, [::]:9000->9000/tcp
67d6c8373158   nginx                               "/docker-entrypoint..." 24 minutes ago Up 24 minutes  0.0.0.0:8081->80/tcp, [::]:8081->80
/tcp
a413b77245b8   alpine                              "sh -c 'while true; ..." 24 minutes ago Up 24 minutes
c11fed6f8634   hello-world                         "/hello"                43 minutes ago Exited (0) 43 minutes ago
07279a7c1933   nextcloud:29                        "/entrypoint.sh apac..." 3 months ago   Up 56 minutes  0.0.0.0:8080->80/tcp, [::]:8080->80
/tcp
be374c0d5c04   mariadb:11                          "docker-entrypoint.s..." 3 months ago   Up 56 minutes  3306/tcp
d9f48799aedb   hello-world                         "/hello"                3 months ago   Exited (0) 3 months ago
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$
```

After using Docker for a while, you often lose track of your containers, making the tool unpredictable. Some ports stop working, and Docker sometimes refuses to create containers because some containers you thought you deleted are still there. To get out of this situation, use the Docker command below, which immediately displays the containers that have exited, crashed, or stopped:

```
docker ps -a
```

The following command can be used for quick scripting or cleanup, as it only returns the container ID:

```
docker ps -q
```

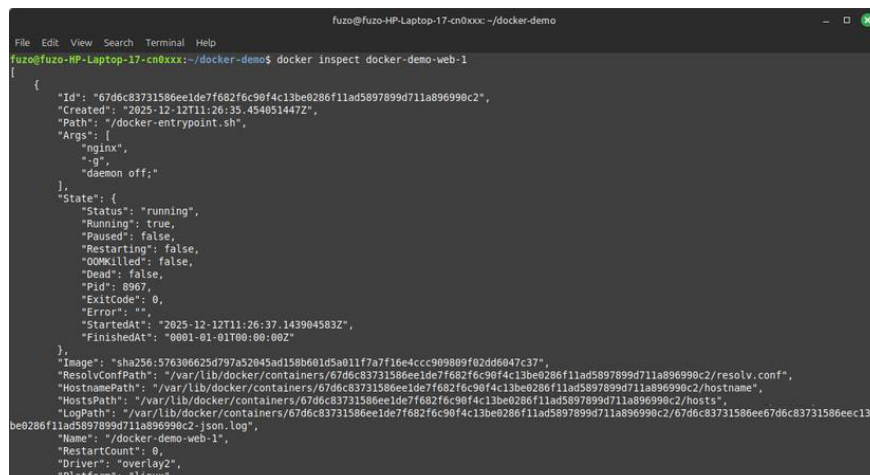
To delete a forgotten container, use the command:

```
docker rm
```

One of the most underrated Docker skills is knowing what exists and what might be silently failing.

## Docker hit

## See everything about containers in detail.



```
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker inspect docker-demo-web-1
[
  {
    "Id": "67d6c83731586ee1de7f682f6c90f4c13be0286f11ad5897899d711a89699c2",
    "Created": "2025-12-12T11:26:35.454651447Z",
    "Path": "/docker-entrypoint.sh",
    "Args": [
      "nginx",
      "-g",
      "daemon off;"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 8967,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2025-12-12T11:26:37.143904583Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:576366625d797a52045ad158b601d5a011f7a7f16e4ccc909889f02dd6047c37",
    "ResolvConfPath": "/var/lib/docker/containers/67d6c83731586ee1de7f682f6c90f4c13be0286f11ad5897899d711a89699c2/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/67d6c83731586ee1de7f682f6c90f4c13be0286f11ad5897899d711a89699c2/hostname",
    "HostsPath": "/var/lib/docker/containers/67d6c83731586ee1de7f682f6c90f4c13be0286f11ad5897899d711a89699c2/hosts",
    "LogPath": "/var/lib/docker/containers/67d6c83731586ee1de7f682f6c90f4c13be0286f11ad5897899d711a89699c2/67d6c83731586ee1de7f682f6c90f4c13be0286f11ad5897899d711a89699c2-json.log",
    "Name": "/docker-demo-web-1",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
  }
]
```

The `docker inspect` command provides clarity when Docker starts displaying connection issues or unexpected behavior. Below is the basic form of the command:

```
docker inspect myapp
```

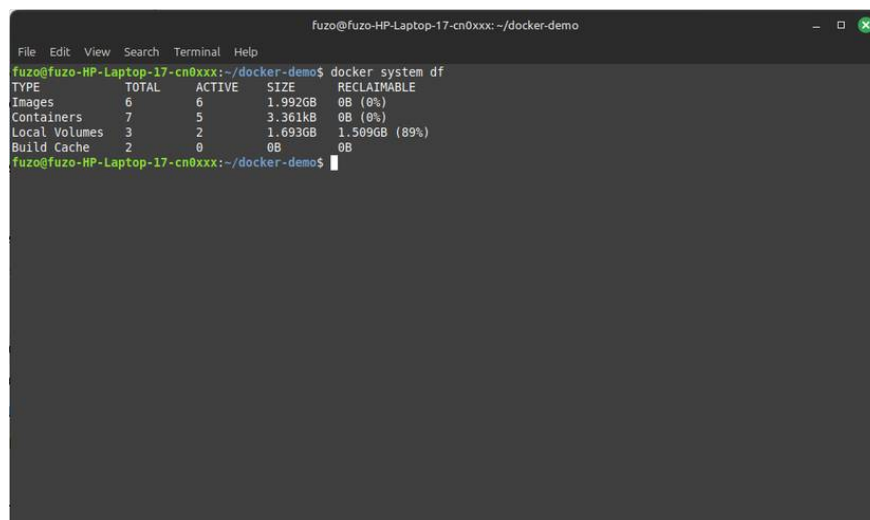
This command displays all the details that Docker sees, including environment variables, network settings, volume, and entry point. You can use the command below to extract a container's IP address when you're curious about a specific component.

```
docker inspect --format='{ {.NetworkSettings.IPAddress} }' myapp
```

By running these centralized commands, you can avoid having to sift through countless JSON pages to debug misconfigurations.

## Docker system prune

Safely clean up unused Docker resources.



```
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker system df
FILE Edit View Search Terminal Help
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$ docker system df
TYPE          TOTAL    ACTIVE   SIZE    RECLAIMABLE
Images        6         6       1.992GB    0B (0%)
Containers    7         5       3.361kB    0B (0%)
Local Volumes 3         2       1.693GB    1.509GB (89%)
Build Cache   2         0         0B         0B
fuzo@fuzo-HP-Laptop-17-cn0xxx:~/docker-demo$
```

When you start using Docker, it silently accumulates unnecessary resources, such as old images, unused network resources, and stopped containers. Over time, this will slow things down if not cleaned up. Start your cleanup process with a safe version:

```
docker system prune
```

This command retains your images but removes unused containers and networks. You can perform a deeper cleanup and remove images as well by running the following command:

```
docker system prune -a
```

Finally, you can run the following command if you need to delete unused volumes that may contain gigabytes of forgotten data:

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

Running the command above will delete unused volumes, but some may be owned by root or other users. Understanding ownership and access permissions in Linux can help make this process safer.

You finished reading the article "**Memorize these 7 commands to stop messing around with Docker containers.**" edited by the [TipsMake](#) team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.