Docker Exit Codes: Troubleshooting Guide

Understand, Diagnose, and Fix Container Failures

Aniruddha Kharve
GitHub: Aniruddhakharve

June 1, 2025

Overview

Docker containers exit with specific codes that help identify what went wrong—or if anything went wrong at all. This guide compiles all common Docker and Linux exit codes to help you debug and fix issues efficiently.

Reference source: Komodor Exit Codes Guide

Standard Exit Codes (0–125)

Exit Code 0 – Success

The container exited cleanly without any errors.

Exit Code 1 – General Error

A general error occurred inside the container (e.g., script or app crash). **Resolution:** Check logs, validate your entrypoint or application logic.

Exit Code 2 – Misuse of Shell Builtins

Shell built-ins were used incorrectly (e.g., syntax error). **Resolution:** Review your shell scripts or Docker entrypoint.

Exit Code 3–125 – Application-Specific

Application-defined exit codes. Interpretation depends on your app. **Resolution:** Consult app documentation and check logs.

Docker Reserved Exit Codes (125–127)

Exit Code 125 – Docker Run Error

The Docker daemon failed to start the container. **Example:** Invalid Docker command, mount issues, permission errors.

Exit Code 126 – Command Invoked Cannot Execute

File exists but is not executable (permissions issue). **Resolution:** Run `chmod +x file.sh`.

Exit Code 127 – Command Not Found

The command does not exist in the container or is not in `$PATH`. **Resolution:** Check your Dockerfile or command path.

Signal-Based Exit Codes (128 + Signal Number)

These exit codes represent Linux signals. Formula: `128 + signal number`.

Exit Code 137 – SIGKILL (9)

Container was forcefully killed (e.g., OOM, `docker kill`). **Resolution:** Increase memory or monitor usage using `docker stats`.

Exit Code 139 – SIGSEGV (11)

Segmentation fault – app accessed restricted memory. **Resolution:** Check application memory handling and native libraries.

Exit Code 143 – SIGTERM (15)

Container terminated gracefully (normal shutdown or scaling down).

Common Linux Signals

- **SIGINT (2):** Interrupt from keyboard (Ctrl+C).
- **SIGTERM (15):** Termination signal for graceful shutdown.
- **SIGKILL (9):** Force kill, cannot be trapped.
- **SIGSEGV (11):** Segmentation fault – invalid memory access.

Best Practices for Troubleshooting

- Use `docker logs <container-id>` to get real-time error logs.
- Always validate entrypoints and ensure executables have correct permissions.
- Use health checks in Dockerfiles to detect errors early.
- Add memory and CPU limits to avoid unexpected OOM kills.
- Monitor container lifecycle via CI/CD pipelines.

Conclusion

Docker exit codes are powerful indicators of what's happening inside your containers. By understanding their meaning and cause, you can diagnose issues faster and maintain reliable, production-ready containerized systems.

Document created by Aniruddha Kharve
github.com/Aniruddhakharve — June 1, 2025