Start containers automatically

By Docker Inc

Published: 2026-02-21 · Archived: 2026-04-05 15:04:16 UTC

Docker provides restart policies to control whether your containers start automatically when they exit, or when Docker restarts. Restart policies start linked containers in the correct order. Docker recommends that you use restart policies, and avoid using process managers to start containers.

Restart policies are different from the `--live-restore` flag of the `dockerd` command. Using `--live-restore` lets you to keep your containers running during a Docker upgrade, though networking and user input are interrupted.

To configure the restart policy for a container, use the `--restart` flag when using the `docker run` command. The value of the `--restart` flag can be any of the following:

Flag	Description
`no`	Don't automatically restart the container. (Default)
`on-failure[:max-retries]`	Restart the container if it exits due to an error, which manifests as a non-zero exit code. Optionally, limit the number of times the Docker daemon attempts to restart the container using the `:max-retries` option. The `on-failure` policy only prompts a restart if the container exits with a failure. It doesn't restart the container if the daemon restarts.
`always`	Always restart the container if it stops. If it's manually stopped, it's restarted only when Docker daemon restarts or the container itself is manually restarted. (See the second bullet listed in restart policy details)
`unless-stopped`	Similar to `always`, except that when the container is stopped (manually or otherwise), it isn't restarted even after Docker daemon restarts.

The following command starts a Redis container and configures it to always restart, unless the container is explicitly stopped, or the daemon restarts.

The following command changes the restart policy for an already running container named `redis`.

The following command ensures all running containers restart.

Restart policy details

Keep the following in mind when using restart policies:

- A restart policy only takes effect after a container starts successfully. In this case, starting successfully means that the container is up for at least 10 seconds and Docker has started monitoring it. This prevents a container which doesn't start at all from going into a restart loop.
- If you manually stop a container, the restart policy is ignored until the Docker daemon restarts or the container is manually restarted. This prevents a restart loop.
- Restart policies only apply to containers. To configure restart policies for Swarm services, see flags related to service restart.

Restarting foreground containers

When you run a container in the foreground, stopping a container causes the attached CLI to exit as well, regardless of the restart policy of the container. This behavior is illustrated in the following example.

1. Create a Dockerfile that prints the numbers 1 to 5 and then exits.
2. Build an image from the Dockerfile.
3. Run a container from the image, specifying `always` for its restart policy.

   The container prints the numbers 1..5 to stdout, and then exits. This causes the attached CLI to exit as well.
4. Running `docker ps` shows that is still running or restarting, thanks to the restart policy. The CLI session has already exited, however. It doesn't survive the initial container exit.
5. You can re-attach your terminal to the container between restarts, using the `docker container attach` command. It's detached again the next time the container exits.

If restart policies don't suit your needs, such as when processes outside Docker depend on Docker containers, you can use a process manager such as systemd or supervisor instead.

> Don't combine Docker restart policies with host-level process managers, as this creates conflicts.

To use a process manager, configure it to start your container or service using the same `docker start` or `docker service` command you would normally use to start the container manually. Consult the documentation for the specific process manager for more details.

Using a process manager inside containers

Process managers can also run within the container to check whether a process is running and starts/restart it if not.

> These aren't Docker-aware, and only monitor operating system processes within the container. Docker doesn't recommend this approach, because it's platform-dependent and may differ between versions of a given Linux distribution.

Source: https://docs.docker.com/config/containers/start-containers-automatically/