Using lifecycle hooks with Compose

- Services lifecycle hooks
  - Post-start hooks
  - Pre-stop hooks
- Reference information

{{< summary-bar feature_name="Compose lifecycle hooks" >}}

Services lifecycle hooks

When Docker Compose runs a container, it uses two elements, ENTRYPOINT and COMMAND, to manage what happens when the container starts and stops.

However, it can sometimes be easier to handle these tasks separately with lifecycle hooks - commands that run right after the container starts or just before it stops.

Lifecycle hooks are particularly useful because they can have special privileges (like running as the root user), even when the container itself runs with lower privileges for security. This means that certain tasks requiring higher permissions can be done without compromising the overall security of the container.

Post-start hooks

Post-start hooks are commands that run after the container has started, but there's no set time for when exactly they will execute. The hook execution timing is not assured during the execution of the container's `entrypoint`.

Because there is no ordering guarantee between the hook and the container's entrypoint, post-start hooks are best suited for tasks that do not need to complete before the application begins running, such as registering the container with an external system.

In the following example, after the container starts, a root-level hook registers the service with an internal service registry. The application does not depend on registration being complete before it starts serving requests.

```
services:
  app:
    image: backend
    user: 1001
    post_start:
      - command: /opt/scripts/register-service.sh
        user: root
```

Pre-stop hooks

Pre-stop hooks are commands that run before the container is stopped by a specific command (like `docker compose down` or stopping it manually with `Ctrl+C`). These hooks won't run if the container stops by itself or gets killed suddenly.

Because the pre-stop hook runs before the stop signal is sent to the container, it is suited for actions that must complete while the application is still fully running.

In the following example, the hook backs up a data file before the container receives the stop signal.

```
services:
  app:
    image: backend
    volumes:
      - data:/data
    pre_stop:
      - command: cp /data/app.db /data/app.db.bak


volumes:
  data: {} # a Docker volume is created with root ownership
```

Reference information

- `post_start`
- `pre_stop`

□□□ #2
□ □□□ □□□ 2026-10-31 19:08:14 UTC
□ □□□ □□□ 2026-10-31 19:08:15 UTC