Compose Develop Specification

Table of contents

Note:

Develop is an optional part of the Compose Specification. It is available with Docker Compose version 2.22.0 and later.

This page defines how Compose behaves to efficiently assist you and defines the development constraints and workflows set by Compose. Only a subset of Compose file services may require a `develop` subsection.

Illustrative example

```
services:
 frontend:
   image: example/webapp
   build: ./webapp
   develop:
     watch:
       # sync static content
       - path: ./webapp/html
         action: sync
         target: /var/www
         ignore:
           - node_modules/

 backend:
   image: example/backend
   build: ./backend
   develop:
     watch:
       # rebuild image and recreate service
       - path: ./backend/src
         action: rebuild
```

Attributes

The `develop` subsection defines configuration options that are applied by Compose to assist you during development of a service with optimized workflows.

watch

The `watch` attribute defines a list of rules that control automatic service updates based on local file changes. `watch` is a sequence, each individual item in the sequence defines a rule to be applied by Compose to monitor source code for changes. For more information, see Use Compose Watch.

action

`action` defines the action to take when changes are detected. If `action` is set to:

- `rebuild`, Compose rebuilds the service image based on the `build` section and recreates the service with the updated image.
- `sync`, Compose keeps the existing service container(s) running, but synchronizes source files with container content according to the `target` attribute.
- `sync+restart`, Compose synchronizes source files with container content according to the `target` attribute, and then restarts the container.

  `sync+restart` attribute is available with Docker Compose version 2.23.0 and later.

ignore

The `ignore` attribute can be used to define a list of patterns for paths to be ignored. Any updated file that matches a pattern, or belongs to a folder that matches a pattern, won't trigger services to be re-created. The syntax is the same as `.dockerignore` file:

- * matches 0 or more characters in a file name.
- ? matches a single character in file name.
- */* matches two nested folders with arbitrary names
- ** matches an arbitrary number of nested folders

If the build context includes a `.dockerignore` file, the patterns in this file is loaded as implicit content for the `ignores` file, and values set in the Compose model are appended.

path

`path` attribute defines the path to source code (relative to the project directory) to monitor for changes. Updates to any file inside the path, which doesn't match any `ignore` rule, triggers the configured action.

target

`target` attribute only applies when `action` is configured for `sync`. Files within `path` with changes are synchronized with container filesystem, so that the latter is always running with up-to-date content.