Extensions

Table of contents

- Example 1
- Example 2
- Example 3
- Example 4
- Informative Historical Notes
- Specifying byte values
- Specifying durations

Extensions can be used to make your Compose file more efficient and easier to maintain.

Use the prefix `x-` as a top-level element to modularize configurations that you want to reuse. Compose ignores any fields that start with `x-`, this is the sole exception where Compose silently ignores unrecognized fields.

Extensions can also be used with anchors and aliases.

They also can be used within any structure in a Compose file where user-defined keys are not expected. Compose uses those to enable experimental features, the same way browsers add support for custom CSS features

Example 1

```
x-custom:
 foo:
   - bar
   - zot

services:
 webapp:
   image: example/webapp
   x-foo: bar

service:
 backend:
   deploy:
     placement:
       x-aws-role: "arn:aws:iam::XXXXXXXXXXXX:role/foo"
       x-aws-region: "eu-west-3"
       x-azure-region: "france-central"
```

Example 2

```
x-env: &env
 environment:
   - CONFIG_KEY
   - EXAMPLE_KEY

services:
 first:
   <<: *env
   image: my-image:latest
 second:
   <<: *env
   image: another-image:latest
```

In this example, the environment variables do not belong to either of the services. Theyâ■■ve been lifted out completely into the `x-env` extension field. This defines a new node which contains the environment field. The `&env` YAML anchor is used so both services can reference the extension fieldâ■■s value as `*env`.

Example 3

```
x-function: &function
labels:
  function: "true"
depends_on:
  - gateway
```

```
networks:
  - functions
deploy:
  placement:
    constraints:
      - 'node.platform.os == linux'
services:
# Node.js gives OS info about the node (Host)
nodeinfo:
  <<: *function
  image: functions/nodeinfo:latest
  environment:
    no_proxy: "gateway"
    https_proxy: $https_proxy
# Uses `cat` to echo back response, fastest function to execute.
echoit:
  <<: *function
  image: functions/alpine:health
  environment:
    fprocess: "cat"
    no_proxy: "gateway"
    https_proxy: $https_proxy
```

The `nodeinfo` and `echoit` services both include the `x-function` extension via the `&function` anchor, then set their specific image and environment.

Example 4

Using YAML merge it is also possible to use multiple extensions and share and override additional attributes for specific needs:

```
x-environment: &default-environment
 FOO: BAR
 ZOT: QUIX
x-keys: &keys
 KEY: VALUE
services:
 frontend:
   image: example/webapp
   environment:
     << : [*default-environment, *keys]
     YET_ANOTHER: VARIABLE
```

Note

YAML merge only applies to mappings, and can't be used with sequences.

In the example above, the environment variables are declared using the `FOO: BAR` mapping syntax, while the sequence syntax `- FOO=BAR` is only valid when no fragments are involved.

Informative Historical Notes

This section is informative. At the time of writing, the following prefixes are known to exist:

Prefix	Vendor/Organization
docker	Docker
kubernetes	Kubernetes

Specifying byte values

Values express a byte value as a string in `{amount}{byte unit}` format: The supported units are `b` (bytes), `k` or `kb` (kilo bytes), `m` or `mb` (mega bytes) and `g` or `gb` (giga bytes).

```
    2b
   1024kb
   2048k
   300m
   1gb
```

Specifying durations

Values express a duration as a string in the form of `{value}{unit}`. The supported units are `us` (microseconds), `ms` (milliseconds), `s` (seconds), `m` (minutes) and `h` (hours). Values can combine multiple values without separator.

```
  10ms
 40s
 1m30s
 1h5m30s20ms
```