Search the Docs (/)

What is Docker? (http://www.docker.com/whatisdocker/) Use Cases (http://www.docker.com/resources/usecases/)

Try It! (http://www.docker.com/tryit/) Browse (https://registry.hub.docker.com)

Log In (https://hub.docker.com/account/login) Sign Up (https://hub.docker.com/account/signup)

About (/) Installation (/installation/ubuntulinux/) User Guide (/userguide/) Docker Hub (/docker-hub/)

Examples (/examples/nodejs_web_app/) Articles (/articles/basics/) Reference (/reference/commandline/cli/)

Contributor Guide (/project/who-written-for/)

# docker-compose.yml reference

Each service defined in `docker-compose.yml` must specify exactly one of `image` or `build`. Other keys are optional, and are analogous to their `docker run` command-line counterparts.

As with `docker run`, options specified in the Dockerfile (e.g., `CMD`, `EXPOSE`, `VOLUME`, `ENV`) are respected by default - you don't need to specify them again in `docker-compose.yml`.

## image

Tag or partial image ID. Can be local or remote - Compose will attempt to pull if it doesn't exist locally.

```
image: ubuntu
image: orchardup/postgresql
image: a4bc65fd
```

## build

Path to a directory containing a Dockerfile. When the value supplied is a relative path, it is interpreted as relative to the location of the yml file itself. This directory is also the build context that is sent to the Docker daemon.

Compose will build and tag it with a generated name, and use that image thereafter.

```
build: /path/to/build/dir
```

## command

Override the default command.

```
command: bundle exec thin -p 3000
```

## links

Link to containers in another service. Either specify both the service name and the link alias (`SERVICE:ALIAS`), or just the service name (which will also be used for the alias).

```
links:
 - db
 - db:database
 - redis
```

An entry with the alias' name will be created in `/etc/hosts` inside containers for this service, e.g:

```
172.17.2.186  db
172.17.2.186  database
172.17.2.187  redis
```

Environment variables will also be created - see the environment variable reference (/compose/env/) for details.

## external_links

Link to containers started outside this `docker-compose.yml` or even outside of Compose, especially for containers that provide shared or common services. `external_links` follow semantics similar to `links` when specifying both the container name and the link alias (`CONTAINER:ALIAS`).

```
external_links:
 - redis_1
 - project_db_1:mysql
 - project_db_1:postgresql
```

## ports

Expose ports. Either specify both ports (`HOST:CONTAINER`), or just the container port (a random host port will be chosen).

> ***Note:*** *When mapping ports in the `HOST:CONTAINER` format, you may experience erroneous results when using a container port lower than 60, because YAML will parse numbers in the format `xx:yy` as sexagesimal (base 60). For this reason, we recommend always explicitly specifying your port mappings as strings.*

```
ports:
 - "3000"
 - "8000:8000"
 - "49100:22"
 - "127.0.0.1:8001:8001"
```

### expose

Expose ports without publishing them to the host machine - they'll only be accessible to linked services. Only the internal port can be specified.

```
expose:
 - "3000"
 - "8000"
```

### volumes

Mount paths as volumes, optionally specifying a path on the host machine (`HOST:CONTAINER`), or an access mode (`HOST:CONTAINER:ro`).

```
volumes:
 - /var/lib/mysql
 - cache/:/tmp/cache
 - ~/configs:/etc/configs/:ro
```

### volumes_from

Mount all of the volumes from another service or container.

```
volumes_from:
 - service_name
 - container_name
```

### environment

Add environment variables. You can use either an array or a dictionary.

Environment variables with only a key are resolved to their values on the machine Compose is running on, which can be helpful for secret or host-specific values.

```
environment:
  RACK_ENV: development
  SESSION_SECRET:

environment:
  - RACK_ENV=development
  - SESSION_SECRET
```

### env_file

Add environment variables from a file. Can be a single value or a list.

If you have specified a Compose file with `docker-compose -f FILE`, paths in `env_file` are relative to the directory that file is in.

Environment variables specified in `environment` override these values.

```
env_file: .env

env_file:
  - ./common.env
  - ./apps/web.env
  - /opt/secrets.env
```

Compose expects each line in an env file to be in `VAR=VAL` format. Lines beginning with `#` (i.e. comments) are ignored, as are blank lines.

```
# Set Rails/Rack environment
RACK_ENV=development
```

### extends

Extend another service, in the current file or another, optionally overriding configuration.

Here's a simple example. Suppose we have 2 files - **common.yml** and **development.yml**. We can use `extends` to define a service in **development.yml** which uses configuration defined in **common.yml**:

**common.yml**

```
webapp:
  build: ./webapp
  environment:
    - DEBUG=false
    - SEND_EMAILS=false
```

**development.yml**

```
web:
  extends:
    file: common.yml
    service: webapp
  ports:
    - "8000:8000"
  links:
    - db
  environment:
    - DEBUG=true
db:
  image: postgres
```

Here, the `web` service in **development.yml** inherits the configuration of the `webapp` service in **common.yml** - the `build` and `environment` keys - and adds `ports` and `links` configuration. It overrides one of the defined environment variables (DEBUG) with a new value, and the other one (SEND_EMAILS) is left untouched.

For more on `extends`, see the tutorial (/compose/extends/#example) and reference (/compose/extends/#reference).

## net

Networking mode. Use the same values as the docker client `--net` parameter.

```
net: "bridge"
net: "none"
net: "container:[name or id]"
net: "host"
```

## pid

```
pid: "host"
```

Sets the PID mode to the host PID mode. This turns on sharing between container and the host operating system the PID address space. Containers launched with this flag will be able to access and manipulate other containers in the bare-metal machine's namespace and vise-versa.

## dns

Custom DNS servers. Can be a single value or a list.

```
dns: 8.8.8.8
dns:
  - 8.8.8.8
  - 9.9.9.9
```

## cap_add, cap_drop

Add or drop container capabilities. See `man 7 capabilities` for a full list.

```
cap_add:
  - ALL

cap_drop:
  - NET_ADMIN
  - SYS_ADMIN
```

## dns_search

Custom DNS search domains. Can be a single value or a list.

```
dns_search: example.com
dns_search:
  - dc1.example.com
  - dc2.example.com
```

## working_dir, entrypoint, user, hostname, domainname, mem_limit, privileged, restart, stdin_open, tty, cpu_shares

Each of these is a single value, analogous to its docker run (https://docs.docker.com/reference/run/) counterpart.

```
cpu_shares: 73

working_dir: /code
entrypoint: /code/entrypoint.sh
user: postgresql

hostname: foo
domainname: foo.com

mem_limit: 1000000000
privileged: true

restart: always

stdin_open: true
tty: true
```

## Compose documentation

- Installing Compose (/compose/install/)
- User guide (/compose/)
- Command line reference (/compose/cli/)
- Compose environment variables (/compose/env/)
- Compose command line completion (/compose/completion/)