Docker Compose Reference

Compose File Version 3 Cheat Sheet

Please remember to check your Docker version along with the file version if you run into any issues. You can find more here:

https://docs.docker.com/compose/compose-file/compose-versioning/

The following commands can be found at https://docs.docker.com/compose/reference/

A list of core commands include the following:

Docker-compose up
#Builds, (re)creates, starts, and attaches to containers for a service.

Docker-compose start
Starts existing containers for a service.

Docker-compose stop
#Stops running containers without removing them. They can be started again with `docker-compose start`.

Docker-compose ps
#This command will list containers

#Docker-compose down

#Stops containers and removes containers, networks, volumes, and images created by `up`.

For a Compose file, we are building a YML or YAML (.yml or .yaml both work) to define our services, networks, and volumes. The default path for a Compose file is ./docker-compose.yml.

Version 3 Example

For examples of version 3 please see:
https://docs.docker.com/compose/compose-file/#compose-file-structure-and-examples

Docker Compose File Reference: https://docs.docker.com/compose/
A compose YML file will look similar to the following as an example:

```yaml
version: "3.8"
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Compose File elements

Also, when building a compose file the following reference can be extremely helpful:

https://docs.docker.com/compose/compose-file/#expose

You can examine the parameters for specific builds, such as:

Secrets Configuration Reference

secrets configuration reference

The top-level `secrets` declaration defines or references secrets that can be granted to the services in this stack. The source of the secret is either `file` or `external`.

- `file` : The secret is created with the contents of the file at the specified path.
- `external` : If set to true, specifies that this secret has already been created. Docker does not attempt to create it, and if it does not exist, a `secret not found` error occurs.
- `name` : The name of the secret object in Docker. This field can be used to reference secrets that contain special characters. The name is used as is and will **not** be scoped with the stack name. Introduced in version 3.5 file format.
- `template_driver` : The name of the templating driver to use, which controls whether and how to evaluate the secret payload as a template. If no driver is set, no templating is used. The only driver currently supported is `golang` , which uses a `golang` . Introduced in version 3.8 file format, and only supported when using `docker stack` .

Networks

networks

Networks to join, referencing entries under the top-level `networks` key.

```
services:
  some-service:
    networks:
     - some-network
     - other-network
```

Restart Policy

RESTART_POLICY

Configures if and how to restart containers when they exit. Replaces `restart` .

- `condition` : One of `none` , `on-failure` or `any` (default: `any`).
- `delay` : How long to wait between restart attempts, specified as a duration (default: 5s).
- `max_attempts` : How many times to attempt to restart a container before giving up (default: never give up). If the restart does not succeed within the configured `window` , this attempt doesn't count toward the configured `max_attempts` value. For example, if `max_attempts` is set to '2', and the restart fails on the first attempt, more than two restarts may be attempted.
- `window` : How long to wait before deciding if a restart has succeeded, specified as a duration (default: decide immediately).