# The Docker Compose file

The Compose file is a YAML file defining

* version (DEPRECATED)
* services (REQUIRED)
* networks
* volumes
* configs
* secrets

The default path for a Compose file is compose.yaml (preferred) or compose.yml in working directory. Compose implementations SHOULD also support docker-compose.yaml and docker-compose.yml for backward compatibility. If both files exist, Compose implementations MUST prefer canonical compose.yaml one.

## configs

Grant access to configs on a per-service basis using the per-service configs configuration. Two different syntax variants are supported.

Note: The config must already exist or be defined in the top-level configs configuration of this stack file, or stack deployment fails.

The **long syntax** provides more granularity in how the config is created within the service’s task containers.

* **source**: The identifier of the config as it is defined in this configuration.
* **target**: The path and name of the file to be mounted in the service’s task containers. Defaults to /<source> if not specified.
* **uid and gid**: The numeric UID or GID that owns the mounted config file within in the service’s task containers. **Both default to 0 on Linux** if not specified. Not supported on Windows.
* **mode**: The permissions for the file that is mounted within the service’s task containers, in octal notation. For instance, 0444 represents world-readable. The **default is 0444**. Configs cannot be writable because they are mounted in a temporary filesystem, so if you set the writable bit, it is ignored. The executable bit can be set.

The following example sets the name of my\_config to redis\_config within the container, sets the mode to 0440 (group-readable) and sets the user and group to 103. The redis service does not have access to the my\_other\_config config.

version: "3.9"

services:

redis:

image: redis:latest

deploy:

replicas: 1

configs:

- source: my\_config

target: /redis\_config

uid: '103'

gid: '103'

mode: 0440

configs:

my\_config:

file: ./my\_config.txt

my\_other\_config:

external: true

You can grant a service access to multiple configs and you can mix long and short syntax. Defining a config does not imply granting a service access to it.

docker config create

Create a config from a file or STDIN

Note: This command works with the Swarm orchestrator.

Usage:

docker config create [OPTIONS] CONFIG file|-

With a file

docker config create my\_config ./config.json

docker config ls

With stdin

printf <config> | docker config create my\_config -

docker config ls

## secrets

Grant access to secrets on a per-service basis using the per-service secrets configuration. Two different syntax variants are supported.

**Note when using docker stack deploy**

The secret must already exist or be [defined in the top-level secrets configuration](https://docs.docker.com/compose/compose-file/compose-file-v3/" \l "secrets-configuration-reference) of the compose file, or stack deployment fails.

The long syntax provides more granularity in how the secret is created within the service’s task containers.

* **source**: The **identifier** of the secret as it is defined in this configuration.
* **target**: The name of the file to be mounted in /run/secrets/ in the service’s task containers. Defaults to source if not specified.
* **uid and gid**: The numeric UID or GID that owns the file within /run/secrets/ in the service’s task containers. Both default to 0 if not specified.
* **mode**: The permissions for the file to be mounted in /run/secrets/ in the service’s task containers, in octal notation. For instance, 0444 represents world-readable. The default in Docker 1.13.1 is 0000, but it is 0444 in newer versions. Secrets cannot be writable because they are mounted in a temporary filesystem, so if you set the writable bit, it is ignored. The executable bit can be set.

The following example sets name of the my\_secret to redis\_secret within the container, sets the mode to 0440 (group-readable) and sets the user and group to 103. The redis service does not have access to the my\_other\_secret secret.

version: "3.9"

services:

redis:

image: redis:latest

deploy:

replicas: 1

secrets:

- source: my\_secret

target: redis\_secret

uid: '103'

gid: '103'

mode: 0440

secrets:

my\_secret:

file: ./my\_secret.txt

my\_other\_secret:

external: true

You can grant a service access to multiple secrets and you can mix long and short syntax. Defining a secret does not imply granting a service access to it.

**Create a secret from a file or STDIN as content**

Note: This command works with the Swarm orchestrator.

Usage:

docker secret create [OPTIONS] SECRET [file|-]

From stdin

printf "my super secret password" | docker secret create my\_secret -

docker secret ls

From a file

docker secret create my\_secret ./secret.json

docker secret ls