Docker Compose Multi-container stacks

START HERE

`docker compose up -d`
Start the stack

`docker compose ps`
List its containers

`docker compose logs -f <service>`
Follow output

`docker compose exec <service> sh`
Shell in a service

`docker compose run --rm <service> sh`
One-off command

`docker compose build`
Build images

`docker compose config`
Check the merged file

`docker compose down`
Stop and remove

CONCEPTS

A project is one Compose file's services, networks and volumes. Commands act on the current project; the folder name is its default name.

Project one compose.yaml
Service definition + replicas
Container running instance

Services reach each other by service name on the project network. Use the container port, not the published one.

Only the plugin syntax is shown: docker compose, with a space. The old standalone docker-compose binary (v1) is retired.

LIFECYCLE

`docker compose up`
Create and start services
Stays attached and stops everything on Ctrl+C.

`docker compose up -d`
Start in the background

`docker compose up --build`
Rebuild images, then start

`docker compose up -d --wait`
Start and wait until healthy
Needs a healthcheck on the services to be meaningful.

`docker compose up -d --force-recreate`
Recreate even if unchanged

`docker compose up -d --no-deps <service>`
Start one service only

`docker compose up -d --scale <service>=<n>`
Run n replicas
Fails if the service publishes a fixed host port.

`docker compose up --watch`
Sync or rebuild on file changes
Needs a develop.watch section in the file.

`docker compose down`
Remove containers and networks

`docker compose down -v`
Also delete the volumes
Destroys named and anonymous volume data.

`docker compose down --remove-orphans`
Drop containers of removed services

LIFECYCLE (cont.)

`docker compose down --rmi local`
Also remove locally built images

`docker compose stop`
Stop without removing

`docker compose restart <service>`
Restart a service

`docker compose start`
Start stopped containers

`docker compose kill [<service>]`
Force stop services

INSPECT

`docker compose ps`
List the project's containers

`docker compose ps -a`
Include stopped containers

`docker compose ps --services`
List service names only

`docker compose logs <service>`
Show a service's output

`docker compose logs -f <service>`
Follow a service's output

`docker compose logs --tail 100 <service>`
Only the last 100 lines

`docker compose logs --since 10m`
Only recent lines

`docker compose logs -t`
Add timestamps

`docker compose images`
List images in use

`docker compose port <service> <port>`
Show the published host port

`docker compose ls`
List all projects on this host

`docker compose version`
Show the Compose version

RUN COMMANDS

`docker compose exec <service> sh`
Shell inside a running service

`docker compose exec -T <service> <cmd>`
No TTY, for scripts and pipes

`docker compose run --rm <service> sh`
One-off container, removed after
Overrides the service command; ports are not published unless you add -P or -p.

RUN COMMANDS (cont.)

`docker compose run --no-deps --rm <service> <cmd>`
Skip starting dependencies

`docker compose cp <service>:<path> <dir>`
Copy files out of a service
Also works in the other direction.

BUILD AND IMAGES

`docker compose build`
Build all services with a build key

`docker compose build <service>`
Build one service

`docker compose build --no-cache`
Ignore the build cache

`docker compose build --pull`
Fetch newer base images

`docker compose pull`
Pull the services' images

`docker compose push`
Push the services' images

FILES AND CONFIG

`docker compose config`
Print the merged, resolved file
Shows interpolated variables, so it catches typos.

`docker compose config -q`
Validate only, print nothing

`docker compose -f <file> up`
Use a specific Compose file

`docker compose -f <a> -f <b> up`
Merge files, later ones win
compose.override.yaml is merged automatically when no -f is given.

`docker compose -p <name> up`
Set the project name

`docker compose --env-file <f> up`
Read variables from another file
A .env file in the project folder is read by default.

`docker compose --profile <p> up`
Enable a profile

Variable syntax in the file

Form	Meaning
`${VAR}`	Value of VAR, empty if unset
`${VAR:-x}`	x if unset or empty
`${VAR:?msg}`	Fail with msg if unset
`$$`	A literal dollar sign

COMPOSE.YAML

Service keys

Key	Use
`image`	Image to run
`build`	Build context or object
`command`	Override the command
`ports`	Publish "host:ctr"
`environment`	Variables as map or list
`env_file`	Load variables from files
`volumes`	Mounts and volumes
`depends_on`	Start order, optional health
`restart`	Restart policy
`healthcheck`	Readiness test
`profiles`	Start only when enabled

Top-level keys

Key	Use
`name`	Project name
`services`	The containers
`volumes`	Named volumes
`networks`	Custom networks
`configs / secrets`	Files mounted into services

Minimal file

```
services:
  web:
    build: .
    ports: ["8080:80"]
    depends_on:
      db: { condition:
service_healthy }
  db:
    image: postgres:17
    volumes: ["dbdata:/var/lib/
postgresql/data"]
volumes:
  dbdata:
```

A top-level version: key is obsolete; Compose ignores it and warns.

RECIPES

Fresh start after changes

```
docker compose down
docker compose build --pull
docker compose up -d --wait
docker compose logs -f --tail
50
```

Dev and prod override files

```
# compose.yaml holds the shared
base
docker compose -f compose.yaml
-f compose.prod.yaml config
docker compose -f compose.yaml
-f compose.prod.yaml up -d
```

Optional service via profile

```
# in the file: profiles:
[debug]
docker compose --profile debug
up -d
COMPOSE_PROFILES=debug docker
compose ps
```

Database dump from a service

```
docker compose exec -T db
pg_dump -U <user> <db> >
dump.sql
docker compose exec -T db psql
-U <user> <db> < dump.sql
```