Docker Compose

Multi-container stacks

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.

LIFECYCLE

`docker compose up`
Create and start services

`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

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

`docker compose down`
Remove containers and networks

`docker compose down -v`
Also delete the volumes

`docker compose stop`
Stop without removing

`docker compose restart <service>`
Restart a service

INSPECT

`docker compose ps`
List the project's containers

`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

INSPECT (cont.)

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

RUN COMMANDS

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

`docker compose run --rm <service> sh`
One-off container, removed after

BUILD AND IMAGES

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

`docker compose pull`
Pull the services' images

FILES AND CONFIG

`docker compose config`
Print the merged, resolved file

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

`docker compose -f <a> -f <b> up`
Merge files, later ones win

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

`docker compose --env-file <f> up`
Read variables from another file

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

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