### What is Docker Compose ??

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

Compose works in all environments: production, staging, development, testing, as well as CI workflows.

Alternate way of Docker Run command in more organised way

### Features of Docker Compose

* Multiple isolated environments on a single host
* Preserve volume data when containers are created
* Only recreate containers that have changed
* Variables and moving a composition between environments

### Install Docker Compose

**Run this command to download the current stable release of Docker Compose:**

* $ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

**Apply executable permissions to the binary**

* $ sudo chmod +x /usr/local/bin/docker-compose

**Test The Installation**

* $ docker-compose --version
* docker-compose version 1.29.2, build 1110ad01

### Docker Compose File Structure

The Compose file is a YAML file defining services, networks and volumes.

The default path for a Compose file is ./docker-compose.yml

**Tip**: You can use either a .yml or .yaml extension for this file. They both work.

* A service definition contains configuration that is applied to each container started for that service, much like passing command-line parameters to docker run.
* Likewise, network and volume definitions are analogous to docker network create and docker volume create.
* As with docker run, options specified in the Dockerfile, such as CMD, EXPOSE, VOLUME, ENV, are respected by default - you don’t need to specify them again in docker-compose.yml

**Example #1 Build Docker Images**

Build can be specified either as a string containing a path to the build context:

$ vim docker-compose.yml

version: "3.9" #Mandatory

services: #Mandatory Where u difine multiple contianers

webapp: # Any name for identification

image: Image\_Name\_youWant:tag

container\_name: test\_Name

build: . #represents dockerfile in present directory

$ docker-compose build

What if you have different dockerfile Name ?

build:

context: .

dockerfile: Dockerfile2

What if you have the dockerfile in different folder ?

build:

context: /path/

dockerfile: Dockerfile2

args:

buildno: 1

**Fields Under Build**

**context** : Either a path to a directory containing a Dockerfile, or a url to a git repository.

**dockerfile**: Alternate file to build with. A build path must also be specified in context

**args**: Add build arguments, which are environment variables accessible only during the build process.

args:

buildno: 1

gitcommithash: cdc3b19

**cache\_from**: A list of images that the engine uses for cache resolution

cache\_from:

- alpine:latest

- corp/web\_app:3.14

**labels**: Add metadata to the resulting image using [Docker labels](https://docs.docker.com/config/labels-custom-metadata/). You can use either an array or a dictionary.

labels:

com.example.description: "Accounting webapp"

com.example.department: "Finance"

**network**: Assign the network to the container

network: custom\_network\_1

**container\_name**: Specify a custom container name, rather than a generated default name.

container\_name: my-web-container

**entrypoint**: Override the default entrypoint.

entrypoint: /code/entrypoint.sh or

entrypoint: ["php", "-d", "memory\_limit=-1", "vendor/bin/phpunit"]

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

environment:

RACK\_ENV: development

SHOW: 'true'

SESSION\_SECRET:

**env\_file:** Add environment variables from a file. Can be a single value or a list.

env\_file: .env

env\_file:

- ./common.env

- ./apps/web.env

- /opt/runtime\_opts.env

**Expose**:

expose:

- "3000"

- "8000"

HealthCheck

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost"]

interval: 1m30s

timeout: 10s

retries: 3

start\_period: 40s

Image

image: ubuntu:18.04

Logging

logging:

driver: syslog

**#2 Example**

**NETWORKS :: docker network create**

version: "3.9"

services:

web:

image: "nginx:alpine"

networks:

- new

worker:

image: "my-worker-image:latest"

networks:

- legacy

networks:

new:

legacy:

**#3 Example**

**VOLUME :: docker volume create**

version: "3.9"

services:

db:

image: db

volumes:

- data-volume:/var/lib/db

backup:

image: backup-service

volumes:

- data-volume:/var/lib/backup/data

volumes:

data-volume:

### Docker Compose commands

Build or rebuild services

$ docker compose build [SERVICE...]

Creates containers for a service.

$ docker compose create [SERVICE...]

Stop and remove containers, networks

$ docker compose down

Create and start containers

$ docker compose up [SERVICE...]

$ docker composer up -d