Resumo do Projeto Inception - 42

# 📌 Objetivo do Projeto

O projeto Inception visa criar uma stack LEMP (Linux, Nginx, MariaDB, PHP) usando Docker Compose. Deves configurar serviços como WordPress e MariaDB, aplicar certificados SSL e garantir persistência de dados através de volumes. O objetivo é compreender e aplicar os conceitos de virtualização com Docker em containers interdependentes.

# 🛠️ O Que Deves Fazer

- Criar os Dockerfiles e configurações necessárias para Nginx, MariaDB e WordPress.  
- Usar volumes para persistência de dados em MariaDB e WordPress.  
- Garantir o uso de uma rede personalizada.  
- Usar certificados SSL autoassinados para Nginx.  
- Garantir que os serviços arrancam corretamente com `docker compose up`.  
- Criar um Makefile para automatizar tarefas.  
  
  
  
o que e um entrypoint?

Um ENTRYPOINT no contexto do Docker é uma instrução no Dockerfile que define o comando principal que será executado quando um container for iniciado a partir daquela imagem.

### 🔧 Diferença entre ENTRYPOINT e CMD

| Instrução | Função Principal |
| --- | --- |
| ENTRYPOINT | Define o processo principal que sempre será executado |
| CMD | Fornece argumentos default para o ENTRYPOINT (ou define o comando, se não houver ENTRYPOINT) |

# 🔧 Comandos Docker: O que Usar vs Evitar

| Ordem | Comando | Deve Usar? | Fase | Finalidade / Notas |
| --- | --- | --- | --- | --- |
| 1️⃣ | docker compose up --build | ✅ | 🔨 Build inicial | Constrói imagens e sobe containers |
| 2️⃣ | docker-compose build | ✅ | 📦 Build sem subir | Reconstrói imagens após editar Dockerfile |
| 3️⃣ | docker-compose up | ✅ | 🚀 Subir sem rebuild | Sobe containers já construídos |
| 4️⃣ | docker compose ps | ✅ | 📋 Verificação | Lista containers do projeto atual |
| 5️⃣ | docker ps | ✅ | 📋 Verificação | Lista todos os containers ativos |
| 6️⃣ | docker compose -p nome ps | ✅ | 📋 Verificação | Lista containers com mesmo nome de projeto |
| 7️⃣ | docker exec -it mariadb /bin/bash | ✅ | 🐚 Acesso | Entra no container mariadb |
| 8️⃣ | mysql -u root -p | ✅ | 🐬 MySQL | Para interagir com a base de dados |
| 9️⃣ | SHOW DATABASES; USE dbsite; ... | ✅ | 📊 Verificação | Consultas SQL úteis |
| 🔟 | docker exec -it nginx bash | ✅ | 🌐 NGINX | Entra no container nginx |
| 11 | ls -l /etc/nginx/certs | ✅ | 🔐 Certificados | Verifica montagem SSL |
| 12 | apt update && apt install -y tree | ✅ | 🌲 Debug | Instalar tree dentro do container |
| 13 | tree /etc/nginx ; tree /usr | ✅ | 📂 Estrutura | Visualiza diretórios |
| 14 | docker volume ls | ✅ | 📋 Volumes | Lista volumes Docker |
| 15 | docker volume inspect nome | ✅ | 🔍 Debug | Ver info detalhada do volume |
| 16 | docker image inspect nome | ✅ | 🔍 Debug | Info da imagem |
| 17 | docker network ls | ✅ | 📡 Redes | Lista redes Docker |
| 18 | docker network inspect nome | ✅ | 🔍 Redes | Info da rede |
| 19 | docker-compose build wordpress | ✅ | 📦 Rebuild parcial | Só recompila o serviço WordPress |
| 20 | docker exec -it wordpress bash | ✅ | 🚪 Acesso WP | Entra no container WordPress |
| 21 | docker compose down | ✅ | 🛑 Fechar | Para e remove containers, redes e volumes do projeto |
| 22 | ls -alR | ✅ | 📁 Estrutura | Lista recursiva de ficheiros |

# 📝 Comandos por Etapa de Avaliação

### 🔨 Build inicial

* Comando: docker compose up --build  
  Constrói todas as imagens e inicia os containers pela primeira vez.

### 📦 Build sem subir

* Comando: docker compose build  
  Apenas recompila as imagens (útil após alterar Dockerfile).

### 🚀 Subir containers após build

* Comando: docker compose up  
  Sobe os serviços usando as imagens já construídas.

### 📋 Verificar estado dos containers

* Comando: docker ps  
  Mostra todos os containers ativos (de todos os projetos).
* Comando: docker compose ps  
  Mostra apenas os containers do projeto atual.
* Comando: docker compose -p <nome> ps  
  Mostra os containers de um projeto com nome específico.

### 📡 Redes

* Comando: docker network ls  
  Lista todas as redes Docker ativas.
* Comando: docker network inspect <nome\_da\_rede>  
  Inspeciona os detalhes de uma rede específica.

### 💾 Volumes

* Comando: docker volume ls  
  Lista todos os volumes ativos.
* Comando: docker volume inspect <nome\_do\_volume>  
  Mostra detalhes do volume (conteúdo, mounting, etc.).

### 🖼️ Imagens

* Comando: docker image inspect <nome\_da\_imagem>  
  Mostra os detalhes da imagem (layers, metadata, etc.).

### 🐚 Aceder ao container MariaDB

* Comando: docker exec -it mariadb /bin/bash  
  Abre um terminal dentro do container MariaDB.

### 🐬 Interagir com a base de dados MySQL

* Comando: mysql -u root -p  
  A partir do container, entra no cliente MySQL.

### 📊 Consultar base de dados

* Comandos:
* SHOW DATABASES;
* USE dbsite;
* SHOW TABLES;
* SELECT \* FROM wp\_users;
* SELECT \* FROM wp\_comments;  
    
  Para verificar a estrutura e os dados da base.

### 🌐 Aceder ao container NGINX

* Comando: docker exec -it nginx bash  
  Permite inspecionar configurações dentro do container.

### 🔐 Verificar certificados SSL

* Comando: ls -l /etc/nginx/certs  
  Verifica se os certificados estão montados corretamente.

### 🌲 Instalar e usar tree dentro do container

* Instalar:  
  apt update && apt install -y tree
* Ver estrutura de diretórios:  
  tree /etc/nginx ; tree /usr  
  Ajuda a visualizar os ficheiros de configuração e código.

### 📦 Build específico do serviço WordPress

* Comando: docker compose build wordpress  
  Recompila apenas o serviço wordpress.

### 🚪 Aceder ao container WordPress

* Comando: docker exec -it wordpress bash  
  Permite instalar pacotes ou navegar dentro do container.

### 🔁 Repetir up após alterações

* Comando: docker compose up  
  Sobe novamente os serviços após rebuild ou alterações.

### 📁 Listar estrutura de ficheiros

* Comando: ls -alR  
  Lista recursivamente todos os ficheiros e pastas.

# ✅ Conclusão

• Usa comandos `docker compose` sempre que possível — evita `docker` puro quando se trata do teu projeto.  
• Evita comandos destrutivos como `docker volume rm` ou `docker rmi`, especialmente durante a avaliação.  
• Segue as etapas de validação com comandos bem definidos: up, exec, mysql, SHOW, etc.

Dockerized WordPress Setup with NGINX and MariaDB

This document provides the configuration files and scripts for setting up a WordPress site using Docker, with NGINX as the web server, MariaDB as the database, and PHP-FPM for processing PHP requests. It includes line-by-line commentary in English for each file, explaining the purpose and functionality of each command or setting. Additionally, it explains what Docker and Docker Compose are, their purposes, and how they differ from virtual machines, providing context for the containerized setup. The setup runs WordPress securely over HTTPS, with MariaDB handling the database, NGINX serving web content, and PHP-FPM managing PHP execution.

Overview of Docker and Docker Compose

What is Docker?

Docker is a platform that uses containerization to package applications and their dependencies into lightweight, portable units called containers. Containers run on the host operating system, sharing its kernel but isolating the application’s environment (e.g., libraries, configurations).

Purpose:

Ensures consistent application behavior across development, testing, and production.

Reduces resource overhead compared to virtual machines by avoiding a full guest OS.

Supports microservices by running multiple isolated containers on a single host.

Enhances portability across systems (e.g., local machines, cloud providers).

How it Works: Docker uses the Docker Engine to manage containers, images (blueprints for containers), and networks. Containers leverage OS-level virtualization (e.g., Linux namespaces and cgroups) for isolation.

What is Docker Compose?

Docker Compose is a tool for defining and managing multi-container Docker applications using a YAML file (docker-compose.yml). It orchestrates containers that form an application stack, such as NGINX, MariaDB, and WordPress in this setup.

Purpose:

Simplifies configuration of multiple containers, including networking and dependencies.

Streamlines development and testing with a single command (docker-compose up).

Manages container lifecycles and inter-container communication.

How it Works: The docker-compose.yml file defines services (containers), networks, and volumes. Docker Compose uses this file to build, run, and link containers.

Docker vs. Virtual Machines

Docker containers and virtual machines (VMs) both provide isolated environments but differ in architecture and use cases:

Docker Containers:

Share the host OS kernel, isolating only the application and its dependencies.

Lightweight (MBs of memory/disk), with near-native performance and fast startup (seconds).

Ideal for microservices, CI/CD, and scalable applications.

Less secure due to process-level isolation.

Virtual Machines:

Run a full guest OS on a hypervisor, emulating hardware.

Resource-heavy (GBs for guest OS), with slower startup (minutes) and higher overhead.

Suited for running different OSes or high-security workloads.

Stronger isolation via full OS separation.

Relevance to This Setup: This WordPress setup uses Docker containers for NGINX, MariaDB, and WordPress, managed by Docker Compose, to achieve a lightweight, portable, and consistent environment. VMs would be overkill, requiring more resources and slower deployment.

NGINX

NGINX Dockerfile

This Dockerfile builds an NGINX container with SSL support.

dockerfile

# Use Debian Bullseye as the base image for the container

FROM debian:bullseye

# Declare build-time arguments for SSL certificate paths and subject fields

ARG CRED\_PATH CRED\_CERT CRED\_KEY COUNTRY STATE LOCALITY ORGANIZATION ORG\_UNIT COMMON\_NAME

# Update package lists, upgrade installed packages, and install NGINX, OpenSSL, and gettext-base

RUN apt update && apt upgrade -y && apt install -y nginx openssl gettext-base

# Create a directory for SSL certificates using the CRED\_PATH variable

RUN mkdir -p ${CRED\_PATH} #Substituted Added line RUN mkdir -p /etc/nginx/certs

# Generate a self-signed SSL certificate valid for 365 days with a 2048-bit RSA key

RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ${CRED\_PATH}/${CRED\_KEY} -out ${CRED\_PATH}/${CRED\_CERT} -subj "/C=${COUNTRY}/ST=${STATE}/L=${LOCALITY}/O=${ORGANIZATION}/OU=${ORG\_UNIT}/CN=${COMMON\_NAME}"

# Copy the default NGINX configuration to a temporary file

COPY default ./tmp/nginx.conf

# Substitute environment variables in the NGINX config and write to sites-available

RUN envsubst '$CRED\_PATH $CRED\_KEY $CRED\_CERT $COMMON\_NAME' < /tmp/nginx.conf > /etc/nginx/sites-available/default

# Set NGINX to run in the foreground as the container's main process

ENTRYPOINT ["nginx", "-g", "daemon off;"]

NGINX Default Configuration

This configuration sets up NGINX to serve WordPress over HTTPS, handling PHP requests via FastCGI.

nginx

# Introductory comments with links to NGINX documentation for configuration guidance

##

# You should look at the following URL's in order to grasp a solid understanding

# of Nginx configuration files in order to fully unleash the power of Nginx.

# https://www.nginx.com/resources/wiki/start/

# https://www.nginx.com/resources/wiki/start/topics/tutorials/config\_pitfalls/

# https://wiki.debian.org/Nginx/DirectoryStructure

#

# In most cases, administrators will remove this file from sites-enabled/ and

# leave it as reference inside of sites-available where it will continue to be

# updated by the nginx packaging team.

#

# This file will automatically load configuration files provided by other

# applications, such as Drupal or Wordpress. These applications will be made

# available underneath a path with that package name, such as /drupal8.

#

# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.

##

# Define the default server configuration block

# Default server configuration

#

server {

# Commented-out line to listen on port 80 for HTTP (not used due to SSL focus)

#listen 80 default\_server; #Commented line

# Commented-out line to listen on port 80 for IPv6 (not used)

#listen [::]:80 default\_server; #Commented line

# Configure the server to listen on port 443 for HTTPS with SSL

listen 443 ssl default\_server;

# Configure the server to listen on port 443 for IPv6 with SSL

listen [::]:443 ssl default\_server;

# Specify allowed SSL/TLS protocols (TLSv1.2 and TLSv1.3)

ssl\_protocols TLSv1.2 TLSv1.3;

# Set the path to the SSL certificate file using substituted variables

ssl\_certificate ${CRED\_PATH}/${CRED\_CERT}; #Value before: /etc/nginx/certs/nginx-selfsigned.crt;

# Set the path to the SSL private key file using substituted variables

ssl\_certificate\_key ${CRED\_PATH}/${CRED\_KEY}; #Value before: /etc/nginx/certs/nginx-selfsigned.key;

# Set the document root directory for serving web content

root /var/www/html;

# Define default files to serve, prioritizing index.php for PHP applications

# Add index.php to the list if you are using PHP

index index.php index.html index.htm index.nginx-debian.html; #Value before:index index.html index.htm index.nginx-debian.html;

# Set the server name (e.g., domain) using the COMMON\_NAME variable

server\_name ${COMMON\_NAME}; #Value before: jode-jes.42.fr;

# Define a location block for handling root URL requests

location / {

# Attempt to serve the URI as a file, then directory, or return 404

# First attempt to serve request as file, then

# as directory, then fall back to displaying a 404.

try\_files $uri $uri/ =404;

}

# Define a location block for handling PHP files

# pass PHP scripts to FastCGI server

#

location ~ \.php$ {

# Include FastCGI configuration for PHP processing

include snippets/fastcgi-php.conf;

# Commented-out option for Unix socket FastCGI (not used)

# With php-fpm (or other unix sockets):

# fastcgi\_pass unix:/run/php/php7.4-fpm.sock;

# Commented-out option for TCP socket FastCGI (not used)

# With php-cgi (or other tcp sockets):

#fastcgi\_pass 127.0.0.1:9000;

# Forward PHP requests to the WordPress container’s PHP-FPM on port 9000

fastcgi\_pass wordpress:9000;

}

# Commented-out block to deny access to .htaccess files (not needed for NGINX)

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

# Commented-out example virtual host configuration for another domain

# Virtual Host configuration for example.com

#

# You can move that to a different file under sites-available/ and symlink that

# to sites-enabled/ to enable it.

#

#server {

# listen 80;

# listen [::]:80;

#

# server\_name example.com;

#

# root /var/www/example.com;

# index index.html;

#

# location / {

# try\_files $uri $uri/ =404;

# }

#}

MariaDB

MariaDB Dockerfile

This Dockerfile builds a MariaDB container for the WordPress database.

dockerfile

# Use Debian Bullseye as the base image for the container

FROM debian:bullseye

# Update package lists, upgrade packages, and install MariaDB server

RUN apt update && apt upgrade -y && apt install -y mariadb-server

# Copy the MariaDB server configuration file to the configuration directory

COPY 50-server.cnf /etc/mysql/mariadb.conf.d/.

# Commented-out line for copying an init.sql file (not used)

#Deleted line -> COPY init.sql /etc/mysql/init.sql

#COPY init.sql /etc/mysql/init.sql

# Create a directory for MariaDB runtime files (e.g., PID and socket)

RUN mkdir /run/mysqld

# Copy the MariaDB startup script to the container

COPY tools/mariadb-run.sh .

# Make the startup script executable

RUN chmod +x ./mariadb-run.sh

# Set the startup script as the container’s main process

ENTRYPOINT ["./mariadb-run.sh"]

MariaDB mariadb-run.sh

This script initializes the MariaDB database and starts the server.

bash

# Specify that the script is a Bash script

#!/bin/bash

# Create an SQL command to create a database if it doesn’t exist, writing to init.sql

echo "CREATE DATABASE IF NOT EXISTS $DB\_NAME;" > /etc/mysql/init.sql

# Append an SQL command to create a user with a password, allowing connections from any host

echo "CREATE USER IF NOT EXISTS '$DB\_USER'@'%' IDENTIFIED BY '$DB\_PASSWORD';" >> /etc/mysql/init.sql

# Append an SQL command to grant all privileges to the user on all databases

echo "GRANT ALL PRIVILEGES ON \*.\* TO '$DB\_USER'@'%' IDENTIFIED BY '$DB\_PASSWORD' WITH GRANT OPTION;" >> /etc/mysql/init.sql

# Append an SQL command to reload privilege tables

echo "FLUSH PRIVILEGES;" >> /etc/mysql/init.sql

# Pause for 5 seconds to ensure system readiness

sleep 5

# Initialize the MariaDB data directory and system tables

mysql\_install\_db

# Start the MariaDB server in the foreground

mysqld

MariaDB 50-server.cnf

This configuration customizes MariaDB server settings.

ini

# Introductory comments explaining the configuration groups

#

# These groups are read by MariaDB server.

# Use it for options that only the server (but not clients) should see

# Define a configuration group for the MariaDB server

# this is read by the standalone daemon and embedded servers

[server]

# Specify an SQL file to execute on server startup

init\_file = /etc/mysql/init.sql

# Define a configuration group for the mysqld standalone daemon

# this is only for the mysqld standalone daemon

[mysqld]

# Introductory comments for basic settings

#

# \* Basic Settings

#

# Commented-out default user setting (typically mysql)

#user = mysql

# Set the MariaDB server to run as the root user

user = root

# Set the location of the PID file

pid-file = /run/mysqld/mysqld.pid

# Set the base directory for MariaDB binaries

basedir = /usr

# Set the directory for database storage

datadir = /var/lib/mysql

# Set the directory for temporary files

tmpdir = /tmp

# Commented-out option to skip DNS resolution for faster connections

# Broken reverse DNS slows down connections considerably and name resolve is

# safe to skip if there are no "host by domain name" access grants

#skip-name-resolve

# Configure MariaDB to listen on the mariadb hostname (Docker network alias)

# Instead of skip-networking the default is now to listen only on

# localhost which is more compatible and is not less secure.

bind-address = mariadb

# Commented-out fine-tuning options (e.g., buffer size, connections)

#

# \* Fine Tuning

#

#key\_buffer\_size = 128M

#max\_allowed\_packet = 1G

#thread\_stack = 192K

#thread\_cache\_size = 8

# This replaces the startup script and checks MyISAM tables if needed

# the first time they are touched

#myisam\_recover\_options = BACKUP

#max\_connections = 100

#table\_cache = 64

# Commented-out logging and replication settings

#

# \* Logging and Replication

#

# Note: The configured log file or its directory need to be created

# and be writable by the mysql user, e.g.:

# $ sudo mkdir -m 2750 /var/log/mysql

# $ sudo chown mysql /var/log/mysql

#

# Both location gets rotated by the cronjob.

# Be aware that this log type is a performance killer.

# Recommend only changing this at runtime for short testing periods if needed!

#general\_log\_file = /var/log/mysql/mysql.log

#general\_log = 1

#

# When running under systemd, error logging goes via stdout/stderr to journald

# and when running legacy init error logging goes to syslog due to

# /etc/mysql/conf.d/mariadb.conf.d/50-mysqld\_safe.cnf

# Enable this if you want to have error logging into a separate file

#log\_error = /var/log/mysql/error.log

# Enable the slow query log to see queries with especially long duration

#log\_slow\_query\_file = /var/log/mysql/mariadb-slow.log

#log\_slow\_query\_time = 10

#log\_slow\_verbosity = query\_plan,explain

#log-queries-not-using-indexes

#log\_slow\_min\_examined\_row\_limit = 1000

#

# The following can be used as easy to replay backup logs or for replication.

# note: if you are setting up a replication slave, see README.Debian about

# other settings you may need to change.

#server-id = 1

#log\_bin = /var/log/mysql/mysql-bin.log

# Set the expiration period for binary logs

expire\_logs\_days = 10

# Commented-out maximum binary log size

#max\_binlog\_size = 100M

# Commented-out SSL/TLS settings

#

# \* SSL/TLS

#

# For documentation, please read

# https://mariadb.com/kb/en/securing-connections-for-client-and-server/

#ssl-ca = /etc/mysql/cacert.pem

#ssl-cert = /etc/mysql/server-cert.pem

#ssl-key = /etc/mysql/server-key.pem

#require-secure-transport = on

# Set the default character set to utf8mb4 for full Unicode support

#

# \* Character sets

#

# MySQL/MariaDB default is Latin1, but in Debian we rather default to the full

# utf8 4-byte character set. See also client.cnf

character-set-server = utf8mb4

# Set the default collation to utf8mb4\_general\_ci (case-insensitive)

collation-server = utf8mb4\_general\_ci

# Commented-out InnoDB settings

#

# \* InnoDB

#

# InnoDB is enabled by default with a 10MB datafile in /var/lib/mysql/.

# Read the manual for more InnoDB related options. There are many!

# Most important is to give InnoDB 80 % of the system RAM for buffer use:

# https://mariadb.com/kb/en/innodb-system-variables/#innodb\_buffer\_pool\_size

#innodb\_buffer\_pool\_size = 8G

# Define a configuration group for the embedded server

# this is only for embedded server

[embedded]

# Define a group for MariaDB-specific options

# This group is only read by MariaDB servers, not by MySQL.

# If you use the same .cnf file for MySQL and MariaDB,

# you can put MariaDB-only options here

[mariadb]

# Define a group for MariaDB-10.11-specific options

# This group is only read by MariaDB-10.11 servers.

# If you use the same .cnf file for MariaDB of different versions,

# use this group for options that older servers don't understand

[mariadb-10.11]

WordPress

WordPress Dockerfile

This Dockerfile builds a WordPress container with PHP-FPM.

dockerfile

# Use Debian Bullseye as the base image for the container

FROM debian:bullseye

# Update package lists, upgrade packages, and install PHP-FPM, MySQLi, and curl

RUN apt update && apt upgrade -y && apt install -y \

php7.4-fpm \

php7.4-mysqli \

curl

# Create a directory for PHP runtime files and set ownership to www-data

RUN mkdir -p /run/php && chown -R www-data:www-data /run/php

# Copy the PHP-FPM pool configuration file

COPY www.conf /etc/php/7.4/fpm/pool.d/.

# Copy the WordPress installation script

COPY ./tools/wp-install.sh .

# Make the installation script executable (note: +X should be +x)

RUN chmod +X wp-install.sh

# Set the installation script as the container’s main process

ENTRYPOINT ["./wp-install.sh"]

WordPress wp-install.sh

This script installs and configures WordPress.

bash

# Specify that the script is a Bash script

#!/bin/bash

# Change to the web root directory for WordPress files

cd /var/www/html

# Check if WP-CLI is not installed

# Instalar wp-cli apenas se não existir

if [ ! -f ./wp-cli.phar ]; then

# Download WP-CLI from its official GitHub repository

curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar

# Make WP-CLI executable

chmod +x wp-cli.phar

fi

# Pause for 5 seconds to ensure MariaDB is ready

# Esperar para garantir que o MariaDB arrancou

sleep 5

# Check if WordPress configuration file does not exist

# Descarregar WordPress apenas se ainda não existir

if [ ! -f ./wp-config.php ]; then

# Download WordPress core files using WP-CLI

./wp-cli.phar core download --allow-root

# Create wp-config.php with database settings

./wp-cli.phar config create \

--dbname=${DB\_NAME} \

--dbuser=${DB\_USER} \

--dbpass=${DB\_PASSWORD} \

--dbhost=${DB\_HOST} \

--allow-root

fi

# Check if WordPress is not installed

# Instalar WordPress apenas se ainda não estiver instalado

if ! ./wp-cli.phar core is-installed --allow-root; then

# Install WordPress with site details

./wp-cli.phar core install \

--url=${WP\_DOMAIN} \

--title=${WP\_TITLE} \

--admin\_user=${WP\_ADMIN\_USER} \

--admin\_password=${WP\_ADMIN\_PASSWORD} \

--admin\_email=${WP\_ADMIN\_EMAIL} \

--allow-root

# Create a subscriber user

./wp-cli.phar user create \

${WP\_GUEST\_USER} \

${WP\_GUEST\_EMAIL} \

--role=subscriber \

--user\_pass=${WP\_GUEST\_PASSWORD} \

--allow-root

fi

# Start PHP-FPM in the foreground

# Iniciar o PHP-FPM no final

php-fpm7.4 -F

WordPress www.conf (PHP-FPM Pool Configuration)

This configuration file customizes the PHP-FPM pool for the WordPress container, defining how PHP processes are managed and how they communicate with NGINX.

ini

; Start a new pool named 'www'.

; the variable $pool can be used in any directive and will be replaced by the

; pool name ('www' here)

[www]

# Defines a PHP-FPM process pool named 'www' for handling PHP requests.

; Per pool prefix

; It only applies on the following directives:

; - 'access.log'

; - 'slowlog'

; - 'listen' (unixsocket)

; - 'chroot'

; - 'chdir'

; - 'php\_values'

; - 'php\_admin\_values'

; When not set, the global prefix (or /usr) applies instead.

; Note: This directive can also be relative to the global prefix.

; Default Value: none

;prefix = /path/to/pools/$pool

# Commented-out directive to set a base path for certain directives; global prefix applies.

; Unix user/group of processes

; Note: The user is mandatory. If the group is not set, the default user's group

; will be used.

user = www-data

# Sets the Unix user for worker processes to www-data for permissions.

group = www-data

# Sets the Unix group for worker processes to www-data for consistency.

; The address on which to accept FastCGI requests.

; Valid syntaxes are:

; 'ip.add.re.ss:port' - to listen on a TCP socket to a specific IPv4 address on

; a specific port;

; '[ip:6:addr:ess]:port' - to listen on a TCP socket to a specific IPv6 address on

; a specific port;

; 'port' - to listen on a TCP socket to all addresses

; (IPv6 and IPv4-mapped) on a specific port;

; '/path/to/unix/socket' - to listen on a unix socket.

; Note: This value is mandatory.

listen = wordpress:9000

# Configures PHP-FPM to listen on the wordpress hostname at port 9000 (TCP socket).

; Set listen(2) backlog.

; Default Value: 511 (-1 on FreeBSD and OpenBSD)

;listen.backlog = 511

# Commented-out directive to set the socket queue size; default (511) applies.

; Set permissions for unix socket, if one is used. In Linux, read/write

; permissions must be set in order to allow connections from a web server. Many

; BSD-derived systems allow connections regardless of permissions. The owner

; and group can be specified either by name or by their numeric IDs.

; Default Values: user and group are set as the running user

; mode is set to 0660

listen.owner = www-data

# Sets the socket owner to www-data (not applicable for TCP socket).

listen.group = www-data

# Sets the socket group to www-data (not applicable for TCP socket).

;listen.mode = 0660

# Commented-out directive to set Unix socket permissions; not used.

; When POSIX Access Control Lists are supported you can set them using

; these options, value is a comma separated list of user/group names.

; When set, listen.owner and listen.group are ignored

;listen.acl\_users =

;listen.acl\_groups =

# Commented-out directives for POSIX ACLs; not used.

; List of addresses (IPv4/IPv6) of FastCGI clients which are allowed to connect.

; Equivalent to the FCGI\_WEB\_SERVER\_ADDRS environment variable in the original

; PHP FCGI (5.2.2+). Makes sense only with a tcp listening socket. Each address

; must be separated by a comma. If this value is left blank, connections will be

; accepted from any ip address.

; Default Value: any

;listen.allowed\_clients = 127.0.0.1

# Commented-out directive to restrict FastCGI clients; accepts all IPs.

; Specify the nice(2) priority to apply to the pool processes (only if set)

; The value can vary from -19 (highest priority) to 20 (lower priority)

; Note: - It will only work if the FPM master process is launched as root

; - The pool processes will inherit the master process priority

; unless it specified otherwise

; Default Value: no set

; process.priority = -19

# Commented-out directive to set process priority; default applies.

; Set the process dumpable flag (PR\_SET\_DUMPABLE prctl) even if the process user

; or group is differrent than the master process user. It allows to create process

; core dump and ptrace the process for the pool user.

; Default Value: no

; process.dumpable = yes

# Commented-out directive to allow core dumps; disabled by default.

; Choose how the process manager will control the number of child processes.

; Possible Values:

; static - a fixed number (pm.max\_children) of child processes;

; dynamic - the number of child processes are set dynamically based on the

; following directives. With this process management, there will be

; always at least 1 children.

; pm.max\_children - the maximum number of children that can

; be alive at the same time.

; pm.start\_servers - the number of children created on startup.

; pm.min\_spare\_servers - the minimum number of children in 'idle'

; state (waiting to process). If the number

; of 'idle' processes is less than this

; number then some children will be created.

; pm.max\_spare\_servers - the maximum number of children in 'idle'

; state (waiting to process). If the number

; of 'idle' processes is greater than this

; number then some children will be killed.

; ondemand - no children are created at startup. Children will be forked when

; new requests will connect. The following parameter are used:

; pm.max\_children - the maximum number of children that

; can be alive at the same time.

; pm.process\_idle\_timeout - The number of seconds after which

; an idle process will be killed.

; Note: This value is mandatory.

pm = dynamic

# Sets the process manager to dynamic, adjusting child processes based on demand.

; The number of child processes to be created when pm is set to 'static' and the

; maximum number of child processes when pm is set to 'dynamic' or 'ondemand'.

; This value sets the limit on the number of simultaneous requests that will be

; served. Equivalent to the ApacheMaxClients directive with mpm\_prefork.

; Equivalent to the PHP\_FCGI\_CHILDREN environment variable in the original PHP

; CGI. The below defaults are based on a server without much resources. Don't

; forget to tweak pm.\* to fit your needs.

; Note: Used when pm is set to 'static', 'dynamic' or 'ondemand'

; Note: This value is mandatory.

pm.max\_children = 5

# Sets the maximum number of child processes to 5 for simultaneous requests.

; The number of child processes created on startup.

; Note: Used only when pm is set to 'dynamic'

; Default Value: (min\_spare\_servers + max\_spare\_servers) / 2

pm.start\_servers = 2

# Sets the number of child processes created at startup to 2.

; The desired minimum number of idle server processes.

; Note: Used only when pm is set to 'dynamic'

; Note: Mandatory when pm is set to 'dynamic'

pm.min\_spare\_servers = 1

# Sets the minimum number of idle processes to 1.

; The desired maximum number of idle server processes.

; Note: Used only when pm is set to 'dynamic'

; Note: Mandatory when pm is set to 'dynamic'

pm.max\_spare\_servers = 3

# Sets the maximum number of idle processes to 3.

; The number of seconds after which an idle process will be killed.

; Note: Used only when pm is set to 'ondemand'

; Default Value: 10s

;pm.process\_idle\_timeout = 10s;

# Commented-out directive for ondemand mode; not applicable.

; The number of requests each child process should execute before respawning.

; This can be useful to work around memory leaks in 3rd party libraries. For

; endless request processing specify '0'. Equivalent to PHP\_FCGI\_MAX\_REQUESTS.

; Default Value: 0

;pm.max\_requests = 500

# Commented-out directive to restart processes after 500 requests; disabled.

; The URI to view the FPM status page. If this value is not set, no URI will be

; recognized as a status page. It shows the following informations:

; pool - the name of the pool;

; process manager - static, dynamic or ondemand;

; start time - the date and time FPM has started;

; start since - number of seconds since FPM has started;

; accepted conn - the number of request accepted by the pool;

; listen queue - the number of request in the queue of pending

; connections (see backlog in listen(2));

; max listen queue - the maximum number of requests in the queue

; of pending connections since FPM has started;

; listen queue len - the size of the socket queue of pending connections;

; idle processes - the number of idle processes;

; active processes - the number of active processes;

; total processes - the number of idle + active processes;

; max active processes - the maximum number of active processes since FPM

; has started;

; max children reached - number of times, the process limit has been reached,

; when pm tries to start more children (works only for

; pm 'dynamic' and 'ondemand');

; Value are updated in real time.

; Example output:

; pool: www

; process manager: static

; start time: 01/Jul/2011:17:53:49 +0200

; start since: 62636

; accepted conn: 190460

; listen queue: 0

; max listen queue: 1

; listen queue len: 42

; idle processes: 4

; active processes: 11

; total processes: 15

; max active processes: 12

; max children reached: 0

;

; By default the status page output is formatted as text/plain. Passing either

; 'html', 'xml' or 'json' in the query string will return the corresponding

; output syntax. Example:

; http://www.foo.bar/status

; http://www.foo.bar/status?json

; http://www.foo.bar/status?html

; http://www.foo.bar/status?xml

;

; By default the status page only outputs short status. Passing 'full' in the

; query string will also return status for each pool process.

; Example:

; http://www.foo.bar/status?full

; http://www.foo.bar/status?json&full

; http://www.foo.bar/status?html&full

; http://www.foo.bar/status?xml&full

; The Full status returns for each process:

; pid - the PID of the process;

; state - the state of the process (Idle, Running, ...);

; start time - the date and time the process has started;

; start since - the number of seconds since the process has started;

; requests - the number of requests the process has served;

; request duration - the duration in µs of the requests;

; request method - the request method (GET, POST, ...);

; request URI - the request URI with the query string;

; content length - the content length of the request (only with POST);

; user - the user (PHP\_AUTH\_USER) (or '-' if not set);

; script - the main script called (or '-' if not set);

; last request cpu - the %cpu the last request consumed

; it's always 0 if the process is not in Idle state

; because CPU calculation is done when the request

; processing has terminated;

; last request memory - the max amount of memory the last request consumed

; it's always 0 if the process is not in Idle state

; because memory calculation is done when the request

; processing has terminated;

; If the process is in Idle state, then informations are related to the

; last request the process has served. Otherwise informations are related to

; the current request being served.

; Example output:

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; pid: 31330

; state: Running

; start time: 01/Jul/2011:17:53:49 +0200

; start since: 63087

; requests: 12808

; request duration: 1250261

; request method: GET

; request URI: /test\_mem.php?N=10000

; content length: 0

; user: -

; script: /home/fat/web/docs/php/test\_mem.php

; last request cpu: 0.00

; last request memory: 0

;

; Note: There is a real-time FPM status monitoring sample web page available

; It's available in: /usr/share/php/7.4/fpm/status.html

;

; Note: The value must start with a leading slash (/). The value can be

; anything, but it may not be a good idea to use the .php extension or it

; may conflict with a real PHP file.

; Default Value: not set

;pm.status\_path = /status

# Commented-out directive to enable a status page; disabled.

; The ping URI to call the monitoring page of FPM. If this value is not set, no

; URI will be recognized as a ping page. This could be used to test from outside

; that FPM is alive and responding, or to

; - create a graph of FPM availability (rrd or such);

; - remove a server from a group if it is not responding (load balancing);

; - trigger alerts for the operating team (24/7).

; Note: The value must start with a leading slash (/). The value can be

; anything, but it may not be a good idea to use the .php extension or it

; may conflict with a real PHP file.

; Default Value: not set

;ping.path = /ping

# Commented-out directive to enable a ping endpoint; disabled.

; This directive may be used to customize the response of a ping request. The

; response is formatted as text/plain with a 200 response code.

; Default Value: pong

;ping.response = pong

# Commented-out directive to customize ping response; default applies if enabled.

; The access log file

; Default: not set

;access.log = log/$pool.access.log

# Commented-out directive to enable access logging; disabled.

; The access log format.

; The following syntax is allowed

; %%: the '%' character

; %C: %CPU used by the request

; it can accept the following format:

; - %{user}C for user CPU only

; - %{system}C for system CPU only

; - %{total}C for user + system CPU (default)

; %d: time taken to serve the request

; it can accept the following format:

; - %{seconds}d (default)

; - %{miliseconds}d

; - %{mili}d

; - %{microseconds}d

; - %{micro}d

; %e: an environment variable (same as $\_ENV or $\_SERVER)

; it must be associated with embraces to specify the name of the env

; variable. Some exemples:

; - server specifics like: %{REQUEST\_METHOD}e or %{SERVER\_PROTOCOL}e

; - HTTP headers like: %{HTTP\_HOST}e or %{HTTP\_USER\_AGENT}e

; %f: script filename

; %l: content-length of the request (for POST request only)

; %m: request method

; %M: peak of memory allocated by PHP

; it can accept the following format:

; - %{bytes}M (default)

; - %{kilobytes}M

; - %{kilo}M

; - %{megabytes}M

; - %{mega}M

; %n: pool name

; %o: output header

; it must be associated with embraces to specify the name of the header:

; - %{Content-Type}o

; - %{X-Powered-By}o

; - %{Transfert-Encoding}o

; - ....

; %p: PID of the child that serviced the request

; %P: PID of the parent of the child that serviced the request

; %q: the query string

; %Q: the '?' character if query string exists

; %r: the request URI (without the query string, see %q and %Q)

; %R: remote IP address

; %s: status (response code)

; %t: server time the request was received

; it can accept a strftime(3) format:

; %d/%b/%Y:%H:%M:%S %z (default)

; The strftime(3) format must be encapsuled in a %{<strftime\_format>}t tag

; e.g. for a ISO8601 formatted timestring, use: %{%Y-%m-%dT%H:%M:%S%z}t

; %T: time the log has been written (the request has finished)

; it can accept a strftime(3) format:

; %d/%b/%Y:%H:%M:%S %z (default)

; The strftime(3) format must be encapsuled in a %{<strftime\_format>}t tag

; e.g. for a ISO8601 formatted timestring, use: %{%Y-%m-%dT%H:%M:%S%z}t

; %u: remote user

;

; Default: "%R - %u %t \"%m %r\" %s"

;access.format = "%R - %u %t \"%m %r%Q%q\" %s %f %{mili}d %{kilo}M %C%%"

# Commented-out directive to customize access log format; default applies if enabled.

; The log file for slow requests

; Default Value: not set

; Note: slowlog is mandatory if request\_slowlog\_timeout is set

;slowlog = log/$pool.log.slow

# Commented-out directive to enable slow request logging; disabled.

; The timeout for serving a single request after which a PHP backtrace will be

; dumped to the 'slowlog' file. A value of '0s' means 'off'.

; Available units: s(econds)(default), m(inutes), h(ours), or d(ays)

; Default Value: 0

;request\_slowlog\_timeout = 0

# Commented-out directive to log slow requests; disabled.

; Depth of slow log stack trace.

; Default Value: 20

;request\_slowlog\_trace\_depth = 20

# Commented-out directive to set slow log stack trace depth; default applies if enabled.

; The timeout for serving a single request after which the worker process will

; be killed. This option should be used when the 'max\_execution\_time' ini option

; does not stop script execution for some reason. A value of '0' means 'off'.

; Available units: s(econds)(default), m(inutes), h(ours), or d(ays)

; Default Value: 0

;request\_terminate\_timeout = 0

# Commented-out directive to kill processes after a timeout; disabled.

; The timeout set by 'request\_terminate\_timeout' ini option is not engaged after

; application calls 'fastcgi\_finish\_request' or when application has finished and

; shutdown functions are being called (registered via register\_shutdown\_function).

; This option will enable timeout limit to be applied unconditionally

; even in such cases.

; Default Value: no

;request\_terminate\_timeout\_track\_finished = no

# Commented-out directive to enforce timeout after certain PHP functions; disabled.

; Set open file descriptor rlimit.

; Default Value: system defined value

;rlimit\_files = 1024

# Commented-out directive to set open file descriptors; system default applies.

; Set max core size rlimit.

; Possible Values: 'unlimited' or an integer greater or equal to 0

; Default Value: system defined value

;rlimit\_core = 0

# Commented-out directive to set core dump size; system default applies.

; Chroot to this directory at the start. This value must be defined as an

; absolute path. When this value is not set, chroot is not used.

; Note: you can prefix with '$prefix' to chroot to the pool prefix or one

; of its subdirectories. If the pool prefix is not set, the global prefix

; will be used instead.

; Note: chrooting is a great security feature and should be used whenever

; possible. However, all PHP paths will be relative to the chroot

; (error\_log, sessions.save\_path, ...).

; Default Value: not set

;chroot =

# Commented-out directive to chroot processes; disabled.

; Chdir to this directory at the start.

; Note: relative path can be used.

; Default Value: current directory or / when chroot

;chdir = /var/www

# Commented-out directive to change working directory; default applies.

; Redirect worker stdout and stderr into main error log. If not set, stdout and

; stderr will be redirected to /dev/null according to FastCGI specs.

; Note: on highloaded environement, this can cause some delay in the page

; process time (several ms).

; Default Value: no

;catch\_workers\_output = yes

# Commented-out directive to log worker output; output goes to /dev/null.

; Decorate worker output with prefix and suffix containing information about

; the child that writes to the log and if stdout or stderr is used as well as

; log level and time. This options is used only if catch\_workers\_output is yes.

; Settings to "no" will output data as written to the stdout or stderr.

; Default value: yes

;decorate\_workers\_output = no

# Commented-out directive to disable log decoration; default applies if enabled.

; Clear environment in FPM workers

; Prevents arbitrary environment variables from reaching FPM worker processes

; by clearing the environment in workers before env vars specified in this

; pool configuration are added.

; Setting to "no" will make all environment variables available to PHP code

; via getenv(), $\_ENV and $\_SERVER.

; Default Value: yes

;clear\_env = no

# Commented-out directive to allow all environment variables; environment cleared.

; Limits the extensions of the main script FPM will allow to parse. This can

; prevent configuration mistakes on the web server side. You should only limit

; FPM to .php extensions to prevent malicious users to use other extensions to

; execute php code.

; Note: set an empty value to allow all extensions.

; Default Value: .php

;security.limit\_extensions = .php .php3 .php4 .php5 .php7

# Commented-out directive to restrict parsed extensions; only .php parsed.

; Pass environment variables like LD\_LIBRARY\_PATH. All $VARIABLEs are taken from

; the current environment.

; Default Value: clean env

;env[HOSTNAME] = $HOSTNAME

;env[PATH] = /usr/local/bin:/usr/bin:/bin

;env[TMP] = /tmp

;env[TMPDIR] = /tmp

;env[TEMP] = /tmp

# Commented-out directives to pass environment variables; none passed.

; Additional php.ini defines, specific to this pool of workers. These settings

; overwrite the values previously defined in the php.ini. The directives are the

; same as the PHP SAPI:

; php\_value/php\_flag - you can set classic ini defines which can

; be overwritten from PHP call 'ini\_set'.

; php\_admin\_value/php\_admin\_flag - these directives won't be overwritten by

; PHP call 'ini\_set'

; For php\_\*flag, valid values are on, off, 1, 0, true, false, yes or no.

;

; Defining 'extension' will load the corresponding shared extension from

; extension\_dir. Defining 'disable\_functions' or 'disable\_classes' will not

; overwrite previously defined php.ini values, but will append the new value

; instead.

;

; Note: path INI options can be relative and will be expanded with the prefix

; (pool, global or /usr)

;

; Default Value: nothing is defined by default except the values in php.ini and

; specified at startup with the -d argument

;php\_admin\_value[sendmail\_path] = /usr/sbin/sendmail -t -i -f www@my.domain.com

;php\_flag[display\_errors] = off

;php\_admin\_value[error\_log] = /var/log/fpm-php.www.log

;php\_admin\_flag[log\_errors] = on

;php\_admin\_value[memory\_limit] = 32M

# Commented-out directives to set PHP configurations; php.ini defaults apply.

Notes

Docker Compose in This Setup: The scripts assume a Docker Compose setup, as evidenced by container names like wordpress and mariadb in network configurations (e.g., fastcgi\_pass wordpress:9000, bind-address = mariadb). A docker-compose.yml file likely defines these services, linking NGINX, MariaDB, and WordPress containers.

Security Considerations: Running MariaDB as root (in 50-server.cnf) is less secure; the mysql user is preferred in production. The self-signed SSL certificate in NGINX is suitable for testing but should be replaced with a trusted certificate (e.g., Let’s Encrypt) for production.

Performance: The PHP-FPM pm = dynamic setting with low max\_children (5) is optimized for low-resource environments. For high-traffic sites, adjust pm.max\_children, pm.min\_spare\_servers, and pm.max\_spare\_servers based on server capacity.

Typo: The chmod +X in the WordPress Dockerfile should be chmod +x for consistency.

This document provides a complete reference for setting up a Dockerized WordPress environment with NGINX, MariaDB, and PHP-FPM, enriched with explanations of Docker, Docker Compose, and their differences from virtual machines. Let me know if you need a sample docker-compose.yml file, further details, or assistance with optimizing the setup!

##################################  
  
  
Below, I provide a line-by-line commentary for the www.conf PHP-FPM configuration file, explaining what each directive or section does. The comments are in English and placed above each line or relevant block for clarity. After the commented file, I integrate this into the existing document for the Dockerized WordPress setup with NGINX and MariaDB, ensuring a cohesive and comprehensive reference.

Line-by-Line Commentary for www.conf

ini

; Start a new pool named 'www'.

; the variable $pool can be used in any directive and will be replaced by the

; pool name ('www' here)

[www]

Defines a PHP-FPM process pool named www. A pool is a group of worker processes that handle PHP requests. The $pool variable can be used in directives to reference the pool name (www).

ini

; Per pool prefix

; It only applies on the following directives:

; - 'access.log'

; - 'slowlog'

; - 'listen' (unixsocket)

; - 'chroot'

; - 'chdir'

; - 'php\_values'

; - 'php\_admin\_values'

; When not set, the global prefix (or /usr) applies instead.

; Note: This directive can also be relative to the global prefix.

; Default Value: none

;prefix = /path/to/pools/$pool

Comments explaining the prefix directive, which sets a base path for certain directives (e.g., logs, sockets). It’s commented out, so the global prefix (typically /usr) applies.

ini

; Unix user/group of processes

; Note: The user is mandatory. If the group is not set, the default user's group

; will be used.

user = www-data

Sets the Unix user for the pool’s worker processes to www-data, a common user for web servers in Debian-based systems, ensuring appropriate permissions.

ini

group = www-data

Sets the Unix group for the pool’s worker processes to www-data, matching the user for consistency and security.

ini

; The address on which to accept FastCGI requests.

; Valid syntaxes are:

; 'ip.add.re.ss:port' - to listen on a TCP socket to a specific IPv4 address on

; a specific port;

; '[ip:6:addr:ess]:port' - to listen on a TCP socket to a specific IPv6 address on

; a specific port;

; 'port' - to listen on a TCP socket to all addresses

; (IPv6 and IPv4-mapped) on a specific port;

; '/path/to/unix/socket' - to listen on a unix socket.

; Note: This value is mandatory.

listen = wordpress:9000

Configures PHP-FPM to listen for FastCGI requests on the wordpress hostname (likely a Docker network alias) at port 9000 using a TCP socket, allowing NGINX to forward PHP requests to this address.

ini

; Set listen(2) backlog.

; Default Value: 511 (-1 on FreeBSD and OpenBSD)

;listen.backlog = 511

Comments explaining the listen.backlog directive, which sets the maximum number of pending connections in the socket queue. Commented out, so the default (511) applies.

ini

; Set permissions for unix socket, if one is used. In Linux, read/write

; permissions must be set in order to allow connections from a web server. Many

; BSD-derived systems allow connections regardless of permissions. The owner

; and group can be specified either by name or by their numeric IDs.

; Default Values: user and group are set as the running user

; mode is set to 0660

listen.owner = www-data

Sets the owner of the socket to www-data. This applies only if a Unix socket is used (not applicable here since a TCP socket is used).

ini

listen.group = www-data

Sets the group of the socket to www-data, aligning with the owner for Unix socket permissions (not applicable here).

ini

;listen.mode = 0660

Commented-out directive to set the permission mode for a Unix socket (e.g., read/write for owner and group). Not used since a TCP socket is configured.

ini

; When POSIX Access Control Lists are supported you can set them using

; these options, value is a comma separated list of user/group names.

; When set, listen.owner and listen.group are ignored

;listen.acl\_users =

;listen.acl\_groups =

Commented-out directives for POSIX ACLs to control socket access. Not used, so listen.owner and listen.group apply (though irrelevant for TCP).

ini

; List of addresses (IPv4/IPv6) of FastCGI clients which are allowed to connect.

; Equivalent to the FCGI\_WEB\_SERVER\_ADDRS environment variable in the original

; PHP FCGI (5.2.2+). Makes sense only with a tcp listening socket. Each address

; must be separated by a comma. If this value is left blank, connections will be

; accepted from any ip address.

; Default Value: any

;listen.allowed\_clients = 127.0.0.1

Commented-out directive to restrict FastCGI clients by IP address. Not set, so connections are accepted from any IP (appropriate for Docker networking).

ini

; Specify the nice(2) priority to apply to the pool processes (only if set)

; The value can vary from -19 (highest priority) to 20 (lower priority)

; Note: - It will only work if the FPM master process is launched as root

; - The pool processes will inherit the master process priority

; unless it specified otherwise

; Default Value: no set

; process.priority = -19

Commented-out directive to set the scheduling priority of worker processes. Not set, so the default system priority applies.

ini

; Set the process dumpable flag (PR\_SET\_DUMPABLE prctl) even if the process user

; or group is differrent than the master process user. It allows to create process

; core dump and ptrace the process for the pool user.

; Default Value: no

; process.dumpable = yes

Commented-out directive to allow core dumps for debugging. Not set, so core dumps are disabled by default.

ini

; Choose how the process manager will control the number of child processes.

; Possible Values:

; static - a fixed number (pm.max\_children) of child processes;

; dynamic - the number of child processes are set dynamically based on the

; following directives. With this process management, there will be

; always at least 1 children.

; pm.max\_children - the maximum number of children that can

; be alive at the same time.

; pm.start\_servers - the number of children created on startup.

; pm.min\_spare\_servers - the minimum number of children in 'idle'

; state (waiting to process). If the number

; of 'idle' processes is less than this

; number then some children will be created.

; pm.max\_spare\_servers - the maximum number of children in 'idle'

; state (waiting to process). If the number

; of 'idle' processes is greater than this

; number then some children will be killed.

; ondemand - no children are created at startup. Children will be forked when

; new requests will connect. The following parameter are used:

; pm.max\_children - the maximum number of children that

; can be alive at the same time.

; pm.process\_idle\_timeout - The number of seconds after which

; an idle process will be killed.

; Note: This value is mandatory.

pm = dynamic

Sets the process manager to dynamic, allowing the number of child processes to adjust based on demand, using settings like pm.max\_children, pm.start\_servers, etc.

ini

; The number of child processes to be created when pm is set to 'static' and the

; maximum number of child processes when pm is set to 'dynamic' or 'ondemand'.

; This value sets the limit on the number of simultaneous requests that will be

; served. Equivalent to the ApacheMaxClients directive with mpm\_prefork.

; Equivalent to the PHP\_FCGI\_CHILDREN environment variable in the original PHP

; CGI. The below defaults are based on a server without much resources. Don't

; forget to tweak pm.\* to fit your needs.

; Note: Used when pm is set to 'static', 'dynamic' or 'ondemand'

; Note: This value is mandatory.

pm.max\_children = 5

Sets the maximum number of child processes to 5, limiting simultaneous PHP requests (suitable for low-resource environments).

ini

; The number of child processes created on startup.

; Note: Used only when pm is set to 'dynamic'

; Default Value: (min\_spare\_servers + max\_spare\_servers) / 2

pm.start\_servers = 2

Sets the number of child processes created at startup to 2 (used since pm is dynamic).

ini

; The desired minimum number of idle server processes.

; Note: Used only when pm is set to 'dynamic'

; Note: Mandatory when pm is set to 'dynamic'

pm.min\_spare\_servers = 1

Sets the minimum number of idle (waiting) child processes to 1, ensuring at least one process is available for new requests.

ini

; The desired maximum number of idle server processes.

; Note: Used only when pm is set to 'dynamic'

; Note: Mandatory when pm is set to 'dynamic'

pm.max\_spare\_servers = 3

Sets the maximum number of idle child processes to 3, allowing PHP-FPM to kill excess idle processes to save resources.

ini

; The number of seconds after which an idle process will be killed.

; Note: Used only when pm is set to 'ondemand'

; Default Value: 10s

;pm.process\_idle\_timeout = 10s;

Commented-out directive for ondemand mode, specifying how long idle processes live. Not applicable since pm is dynamic.

ini

; The number of requests each child process should execute before respawning.

; This can be useful to work around memory leaks in 3rd party libraries. For

; endless request processing specify '0'. Equivalent to PHP\_FCGI\_MAX\_REQUESTS.

; Default Value: 0

;pm.max\_requests = 500

Commented-out directive to restart child processes after 500 requests to prevent memory leaks. Not set, so processes run indefinitely (0).

ini

; The URI to view the FPM status page. If this value is not set, no URI will be

; recognized as a status page. It shows the following informations:

; pool - the name of the pool;

; process manager - static, dynamic or ondemand;

; start time - the date and time FPM has started;

; start since - number of seconds since FPM has started;

; accepted conn - the number of request accepted by the pool;

; listen queue - the number of request in the queue of pending

; connections (see backlog in listen(2));

; max listen queue - the maximum number of requests in the queue

; of pending connections since FPM has started;

; listen queue len - the size of the socket queue of pending connections;

; idle processes - the number of idle processes;

; active processes - the number of active processes;

; total processes - the number of idle + active processes;

; max active processes - the maximum number of active processes since FPM

; has started;

; max children reached - number of times, the process limit has been reached,

; when pm tries to start more children (works only for

; pm 'dynamic' and 'ondemand');

; Value are updated in real time.

; Example output:

; pool: www

; process manager: static

; start time: 01/Jul/2011:17:53:49 +0200

; start since: 62636

; accepted conn: 190460

; listen queue: 0

; max listen queue: 1

; listen queue len: 42

; idle processes: 4

; active processes: 11

; total processes: 15

; max active processes: 12

; max children reached: 0

;

; By default the status page output is formatted as text/plain. Passing either

; 'html', 'xml' or 'json' in the query string will return the corresponding

; output syntax. Example:

; http://www.foo.bar/status

; http://www.foo.bar/status?json

; http://www.foo.bar/status?html

; http://www.foo.bar/status?xml

;

; By default the status page only outputs short status. Passing 'full' in the

; query string will also return status for each pool process.

; Example:

; http://www.foo.bar/status?full

; http://www.foo.bar/status?json&full

; http://www.foo.bar/status?html&full

; http://www.foo.bar/status?xml&full

; The Full status returns for each process:

; pid - the PID of the process;

; state - the state of the process (Idle, Running, ...);

; start time - the date and time the process has started;

; start since - the number of seconds since the process has started;

; requests - the number of requests the process has served;

; request duration - the duration in µs of the requests;

; request method - the request method (GET, POST, ...);

; request URI - the request URI with the query string;

; content length - the content length of the request (only with POST);

; user - the user (PHP\_AUTH\_USER) (or '-' if not set);

; script - the main script called (or '-' if not set);

; last request cpu - the %cpu the last request consumed

; it's always 0 if the process is not in Idle state

; because CPU calculation is done when the request

; processing has terminated;

; last request memory - the max amount of memory the last request consumed

; it's always 0 if the process is not in Idle state

; because memory calculation is done when the request

; processing has terminated;

; If the process is in Idle state, then informations are related to the

; last request the process has served. Otherwise informations are related to

; the current request being served.

; Example output:

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; pid: 31330

; state: Running

; start time: 01/Jul/2011:17:53:49 +0200

; start since: 63087

; requests: 12808

; request duration: 1250261

; request method: GET

; request URI: /test\_mem.php?N=10000

; content length: 0

; user: -

; script: /home/fat/web/docs/php/test\_mem.php

; last request cpu: 0.00

; last request memory: 0

;

; Note: There is a real-time FPM status monitoring sample web page available

; It's available in: /usr/share/php/7.4/fpm/status.html

;

; Note: The value must start with a leading slash (/). The value can be

; anything, but it may not be a good idea to use the .php extension or it

; may conflict with a real PHP file.

; Default Value: not set

;pm.status\_path = /status

Commented-out directive to enable a status page for monitoring PHP-FPM metrics (e.g., active processes, queue length). Not set, so no status page is available.

ini

; The ping URI to call the monitoring page of FPM. If this value is not set, no

; URI will be recognized as a ping page. This could be used to test from outside

; that FPM is alive and responding, or to

; - create a graph of FPM availability (rrd or such);

; - remove a server from a group if it is not responding (load balancing);

; - trigger alerts for the operating team (24/7).

; Note: The value must start with a leading slash (/). The value can be

; anything, but it may not be a good idea to use the .php extension or it

; may conflict with a real PHP file.

; Default Value: not set

;ping.path = /ping

Commented-out directive to enable a ping endpoint for checking PHP-FPM’s availability. Not set, so no ping endpoint is configured.

ini

; This directive may be used to customize the response of a ping request. The

; response is formatted as text/plain with a 200 response code.

; Default Value: pong

;ping.response = pong

Commented-out directive to customize the response for the ping endpoint. Not set, so the default (pong) would apply if enabled.

ini

; The access log file

; Default: not set

;access.log = log/$pool.access.log

Commented-out directive to enable access logging for PHP-FPM requests. Not set, so no access log is generated.

ini

; The access log format.

; The following syntax is allowed

; %%: the '%' character

; %C: %CPU used by the request

; it can accept the following format:

; - %{user}C for user CPU only

; - %{system}C for system CPU only

; - %{total}C for user + system CPU (default)

; %d: time taken to serve the request

; it can accept the following format:

; - %{seconds}d (default)

; - %{miliseconds}d

; - %{mili}d

; - %{microseconds}d

; - %{micro}d

; %e: an environment variable (same as $\_ENV or $\_SERVER)

; it must be associated with embraces to specify the name of the env

; variable. Some exemples:

; - server specifics like: %{REQUEST\_METHOD}e or %{SERVER\_PROTOCOL}e

; - HTTP headers like: %{HTTP\_HOST}e or %{HTTP\_USER\_AGENT}e

; %f: script filename

; %l: content-length of the request (for POST request only)

; %m: request method

; %M: peak of memory allocated by PHP

; it can accept the following format:

; - %{bytes}M (default)

; - %{kilobytes}M

; - %{kilo}M

; - %{megabytes}M

; - %{mega}M

; %n: pool name

; %o: output header

; it must be associated with embraces to specify the name of the header:

; - %{Content-Type}o

; - %{X-Powered-By}o

; - %{Transfert-Encoding}o

; - ....

; %p: PID of the child that serviced the request

; %P: PID of the parent of the child that serviced the request

; %q: the query string

; %Q: the '?' character if query string exists

; %r: the request URI (without the query string, see %q and %Q)

; %R: remote IP address

; %s: status (response code)

; %t: server time the request was received

; it can accept a strftime(3) format:

; %d/%b/%Y:%H:%M:%S %z (default)

; The strftime(3) format must be encapsuled in a %{<strftime\_format>}t tag

; e.g. for a ISO8601 formatted timestring, use: %{%Y-%m-%dT%H:%M:%S%z}t

; %T: time the log has been written (the request has finished)

; it can accept a strftime(3) format:

; %d/%b/%Y:%H:%M:%S %z (default)

; The strftime(3) format must be encapsuled in a %{<strftime\_format>}t tag

; e.g. for a ISO8601 formatted timestring, use: %{%Y-%m-%dT%H:%M:%S%z}t

; %u: remote user

;

; Default: "%R - %u %t \"%m %r\" %s"

;access.format = "%R - %u %t \"%m %r%Q%q\" %s %f %{mili}d %{kilo}M %C%%"

Commented-out directive to customize the format of access logs. Not set, so the default format applies if logging is enabled.

ini

; The log file for slow requests

; Default Value: not set

; Note: slowlog is mandatory if request\_slowlog\_timeout is set

;slowlog = log/$pool.log.slow

Commented-out directive to enable logging of slow PHP requests. Not set, so slow request logging is disabled.

ini

; The timeout for serving a single request after which a PHP backtrace will be

; dumped to the 'slowlog' file. A value of '0s' means 'off'.

; Available units: s(econds)(default), m(inutes), h(ours), or d(ays)

; Default Value: 0

;request\_slowlog\_timeout = 0

Commented-out directive to log requests exceeding a timeout with a backtrace. Not set, so slow request logging is off.

ini

; Depth of slow log stack trace.

; Default Value: 20

;request\_slowlog\_trace\_depth = 20

Commented-out directive to set the stack trace depth for slow request logs. Not set, so the default (20) applies if enabled.

ini

; The timeout for serving a single request after which the worker process will

; be killed. This option should be used when the 'max\_execution\_time' ini option

; does not stop script execution for some reason. A value of '0' means 'off'.

; Available units: s(econds)(default), m(inutes), h(ours), or d(ays)

; Default Value: 0

;request\_terminate\_timeout = 0

Commented-out directive to kill worker processes after a request timeout. Not set, so no timeout is enforced.

ini

; The timeout set by 'request\_terminate\_timeout' ini option is not engaged after

; application calls 'fastcgi\_finish\_request' or when application has finished and

; shutdown functions are being called (registered via register\_shutdown\_function).

; This option will enable timeout limit to be applied unconditionally

; even in such cases.

; Default Value: no

;request\_terminate\_timeout\_track\_finished = no

Commented-out directive to enforce request\_terminate\_timeout even after certain PHP functions. Not set, so the default behavior applies.

ini

; Set open file descriptor rlimit.

; Default Value: system defined value

;rlimit\_files = 1024

Commented-out directive to set the maximum number of open file descriptors. Not set, so the system default applies.

ini

; Set max core size rlimit.

; Possible Values: 'unlimited' or an integer greater or equal to 0

; Default Value: system defined value

;rlimit\_core = 0

Commented-out directive to set the maximum core dump size. Not set, so the system default applies.

ini

; Chroot to this directory at the start. This value must be defined as an

; absolute path. When this value is not set, chroot is not used.

; Note: you can prefix with '$prefix' to chroot to the pool prefix or one

; of its subdirectories. If the pool prefix is not set, the global prefix

; will be used instead.

; Note: chrooting is a great security feature and should be used whenever

; possible. However, all PHP paths will be relative to the chroot

; (error\_log, sessions.save\_path, ...).

; Default Value: not set

;chroot =

Commented-out directive to chroot the worker processes to a specific directory for security. Not set, so chroot is not used.

ini

; Chdir to this directory at the start.

; Note: relative path can be used.

; Default Value: current directory or / when chroot

;chdir = /var/www

Commented-out directive to change the working directory of worker processes. Not set, so the default (current directory) applies.

ini

; Redirect worker stdout and stderr into main error log. If not set, stdout and

; stderr will be redirected to /dev/null according to FastCGI specs.

; Note: on highloaded environement, this can cause some delay in the page

; process time (several ms).

; Default Value: no

;catch\_workers\_output = yes

Commented-out directive to log worker output to the main error log. Not set, so output goes to /dev/null.

ini

; Decorate worker output with prefix and suffix containing information about

; the child that writes to the log and if stdout or stderr is used as well as

; log level and time. This options is used only if catch\_workers\_output is yes.

; Settings to "no" will output data as written to the stdout or stderr.

; Default value: yes

;decorate\_workers\_output = no

Commented-out directive to disable decoration of worker output logs. Not set, so the default (decorated output) applies if enabled.

ini

; Clear environment in FPM workers

; Prevents arbitrary environment variables from reaching FPM worker processes

; by clearing the environment in workers before env vars specified in this

; pool configuration are added.

; Setting to "no" will make all environment variables available to PHP code

; via getenv(), $\_ENV and $\_SERVER.

; Default Value: yes

;clear\_env = no

Commented-out directive to allow all environment variables in PHP. Not set, so the environment is cleared by default.

ini

; Limits the extensions of the main script FPM will allow to parse. This can

; prevent configuration mistakes on the web server side. You should only limit

; FPM to .php extensions to prevent malicious users to use other extensions to

; execute php code.

; Note: set an empty value to allow all extensions.

; Default Value: .php

;security.limit\_extensions = .php .php3 .php4 .php5 .php7

Commented-out directive to restrict PHP-FPM to parsing specific file extensions. Not set, so only .php files are parsed.

ini

; Pass environment variables like LD\_LIBRARY\_PATH. All $VARIABLEs are taken from

; the current environment.

; Default Value: clean env

;env[HOSTNAME] = $HOSTNAME

;env[PATH] = /usr/local/bin:/usr/bin:/bin

;env[TMP] = /tmp

;env[TMPDIR] = /tmp

;env[TEMP] = /tmp

Commented-out directives to pass specific environment variables to PHP. Not set, so no additional variables are passed.

ini

; Additional php.ini defines, specific to this pool of workers. These settings

; overwrite the values previously defined in the php.ini. The directives are the

; same as the PHP SAPI:

; php\_value/php\_flag - you can set classic ini defines which can

; be overwritten from PHP call 'ini\_set'.

; php\_admin\_value/php\_admin\_flag - these directives won't be overwritten by

; PHP call 'ini\_set'

; For php\_\*flag, valid values are on, off, 1, 0, true, false, yes or no.

;

; Defining 'extension' will load the corresponding shared extension from

; extension\_dir. Defining 'disable\_functions' or 'disable\_classes' will not

; overwrite previously defined php.ini values, but will append the new value

; instead.

;

; Note: path INI options can be relative and will be expanded with the prefix

; (pool, global or /usr)

;

; Default Value: nothing is defined by default except the values in php.ini and

; specified at startup with the -d argument

;php\_admin\_value[sendmail\_path] = /usr/sbin/sendmail -t -i -f www@my.domain.com

;php\_flag[display\_errors] = off

;php\_admin\_value[error\_log] = /var/log/fpm-php.www.log

;php\_admin\_flag[log\_errors] = on

;php\_admin\_value[memory\_limit] = 32M

Commented-out directives to set PHP configuration options specific to this pool (e.g., error logging, memory limits). Not set, so php.ini defaults apply.