



Inception

42-School Project



Youssef Agnaou · [Follow](#)

11 min read · Jun 5, 2024

Listen

Share



Inception — 42 School Project

About the Project

This project is a system administration-related one, and in this, I'll document pretty much everything you need to know about the project from start to finish, hope you find this useful 😊

The topics we will discuss in this article are:

- *What is Docker?*
- *What is a Docker Image?*
- *What is a Dockerfile?*
- *What is a Docker Compose?*
- *What is a Container?*
- *What is The Difference Between Container and VM?*
- *What is PID 1?*
- *Is the Daemon Process PID 1? And How Does They Differ From Each Other?*
- *What is WP-CLI?*
- *Why Do You Need to Work With WP-CLI?*
- *What is Redis Cache? And Why Do You Need it in WordPress?*
- *What is FTP? And How Does it Work?*
- *How to Configure FTP? And How to Test If it's Working?*

What is Docker?

Docker is a tool designed to allow you to build, deploy and run applications in an isolated and consistent manner across different machines and operating systems. This process is done using CONTAINERS. which are lightweight virtualized environments that package all the dependencies and code an application needs to run into a single text file, which can run the same way on any machine.

While Docker is primarily used to package and run applications in containers, it is not limited to that use case. Docker can also be used to create and run other types of containers, such as ones for testing, development, or experimentation.

What is a Docker Image?

Docker Image is a lightweight executable package that includes everything the application needs to run, including the code, runtime environment, system tools, libraries, and dependencies.

Although it cannot guarantee error-free performance, as the behavior of an application ultimately depends on many factors beyond the image itself, using Docker can reduce the likelihood of unexpected errors.

Docker Image is built from a **DOCKERFILE**, which is a simple text file that contains a set of instructions for building the image, with each instruction creating a new layer in the image.

What is a Dockerfile?

Dockerfile is that *SIMPLE TEXT FILE* that I mentioned earlier, which contains a set of instructions for building a Docker Image. It specifies the base image to use and then includes a series of commands that automate the process for configuring and building the image, such as installing packages, copying files, and setting environment variables. Each command in the Dockerfile creates a new layer in the image.

Here's an example of a Dockerfile to make things a little bit clear:

```
# This Specifies the base images for the container
# (in this case, it's the 3.14 version of Alpine)
FROM alpine:3.14

# This Run commands in the container shell,
# and installs the specified packages
# (it will install nginx & openssl,
# and will create the directory "/run/nginx" as well)
RUN apk update && \
    apk add nginx openssl && \
    mkdir -p /run/nginx

# This Copies the contents of "./conf/nginx.conf" on the host
# machine to the "/etc/nginx/http.d/"
# directory inside the container
COPY ./conf/nginx.conf /etc/nginx/http.d/default.conf
```

```
# This Specifies the command that will run when the container get started
CMD ["nginx", "-g", "daemon off;"]
```

What is a Docker Compose?

Docker Compose is a powerful tool that simplifies the deployment and management of multi-container Docker applications. It provides several benefits, including simplifying the process of defining related services, volumes for data persistence, and networks for connecting containers. With Docker Compose, you can easily configure each service's settings, including the image to use, the ports to expose, and the environment variables to set...

Overall, Docker Compose streamlines the development process, making it easier for you to build and deliver your applications with greater efficiency and ease.

A Docker Compose has *3 important parts*, which are:

- **Services:** A service is a unit of work in Docker Compose, it has a name, and it defines a container images, a set of environment variables, and a set of ports that are exposed to the host machine. When you run `docker-compose up`, Docker will create a new container for each service in your Compose file.
- **Networks:** A network is a way for containers to communicate with each other. When you create a network in your Compose file, Docker will create a new network that all the other containers in your Compose file will be connected to. This allows containers to communicate with each other without even knowing the IP of each other, just by the name.
- **Volumes:** A volume is a way to store data that is shared between containers. When you create a volume in your Compose file, Docker will create a new volume (a folder in another way) that all the containers have access to. This allows you to share data between the containers without having to copy-paste each and every time you want that data.

Here's an example of a Docker Compose to make things a little bit clear:

```
version: '3'

# All the services that you will work with should be declared under
# the SERVICES section!
services:

  # Name of the first service (for example: nginx)
  nginx:

    # The hostname of the service (will be the same as the service name!)
    hostname: nginx

    # Where the service exist (path) so you can build it
    build:
      context: ./requirements/nginx
      dockerfile: Dockerfile

    # Restart to always keep the service restarting in case of
    # any unexpected errors causing it to go down
    restart: always

    # This line explains itself!!!
    depends_on:
      - wordpress

    # The ports that will be exposed and you will work with
    ports:
      - 443:443

    # The volumes that you will be mounted when the container gets built
    volumes:
      - wordpress:/var/www/html

    # The networks that the container will connect and communicate
    # with the other containers
    networks:
      - web
```

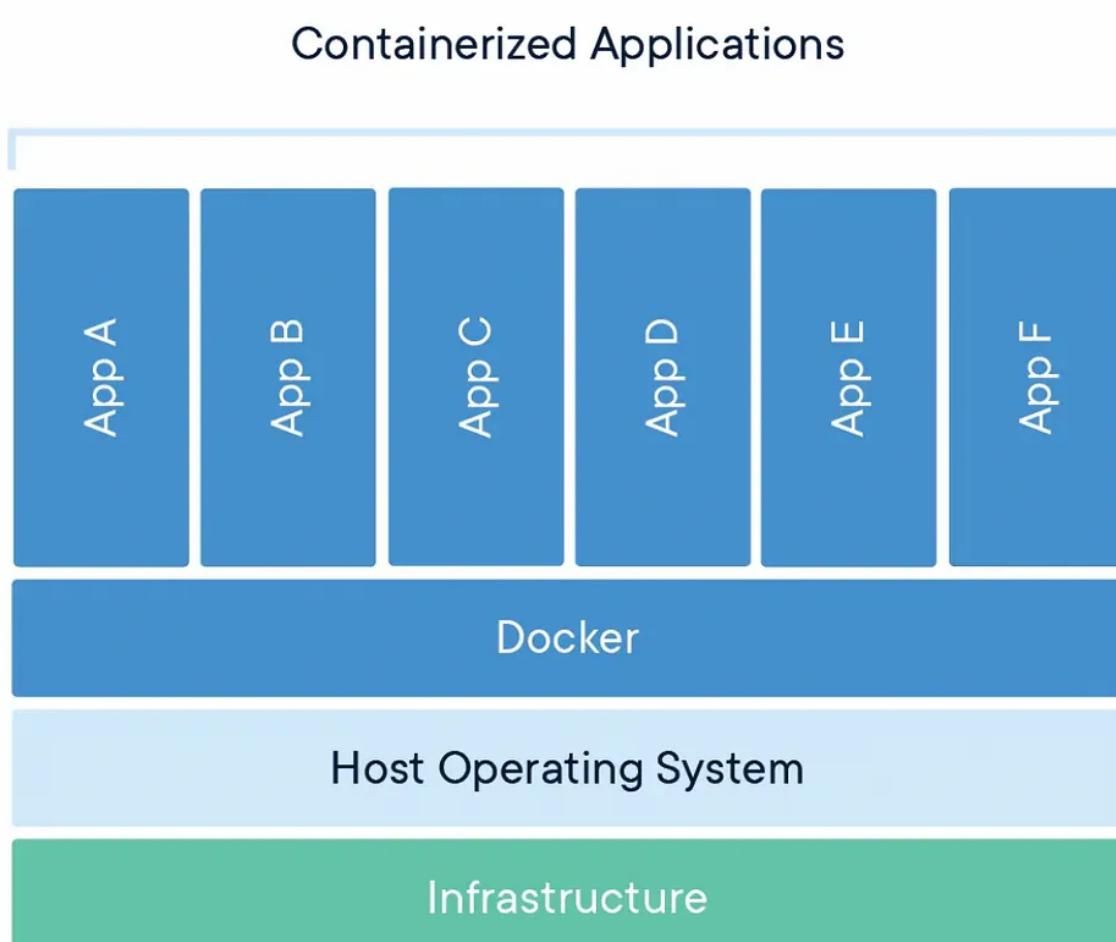
What is a Container?

A Container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

What is The Difference Between Container and VM?

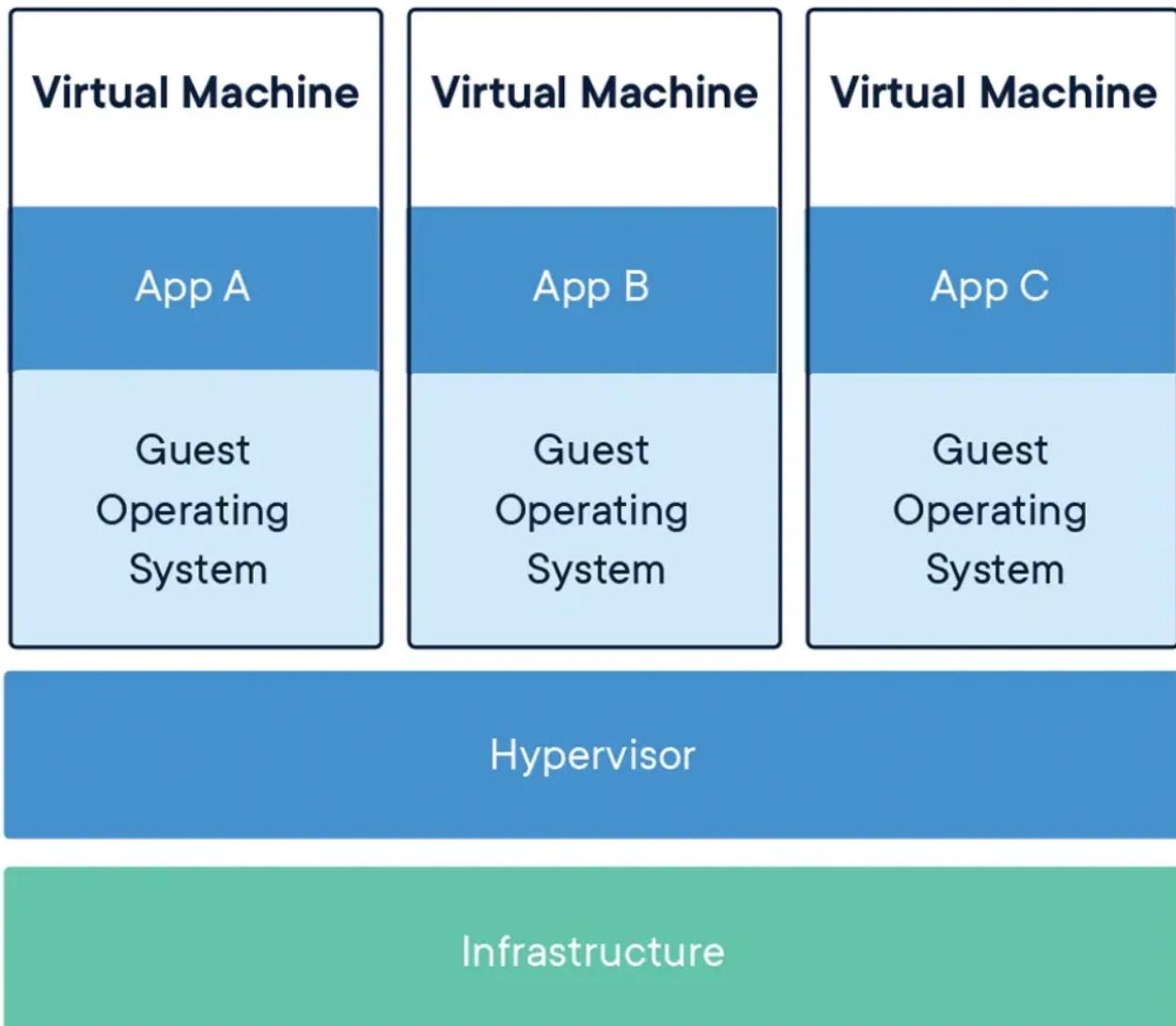
Containers and Virtual machines have similar resource isolation and allocation benefits but function differently because containers virtualize the operating system instead of the hardware. Containers are more portable and efficient.

- **Containers** are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (talks about tens of MBs in size), can handle more applications, and require fewer VMs and Operating Systems.



- **Virtual Machines (VMs)** are an abstraction of physical hardware turning one

server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each machine includes a full copy of an operating system, the application, necessary binaries, and libraries. VMs take more space compared to containers (talking about tens of GBs in size), and they can be slow to boot.



What is PID 1?

In a Docker container, the PID 1 process is a special process that plays an important role in the container's lifecycle. This process is the identifier of the init process, which is the first process that is started when the system boots up, and it is responsible for starting and stopping all of the other processes on the system. And in Docker as well, the init process is responsible for starting and stopping the application that is running in the container.

PID 1 in a Docker container behaves differently from the init process in a normal Unix-based system. (they are NOT the same!)

Is the Daemon Process PID 1? And How Does They Differ From Each Other?

The daemon process is NOT the PID 1, the daemon process is a background process that runs continuously on a system and performs a specific task. In contrast, PID 1 is the first process that the kernel starts in a Unix-based system and plays a special role in the system.

What is WP-CLI?

WP-CLI is the command line interface for WordPress. It is a tool that allows you to interact with your WordPress site from the command line, it is used for a lot of purposes, such as automating tasks, debugging problems, installing/removing plugins along side with themes, managing users and roles, exporting/importing data, run databases queries, and so much more...

Why Do You Need to Work With WP-CLI?

Since it's a tool that allows you to interact directly with your WordPress site from the terminal, it can save you a lot of time that will take you to (for example) installing a plugin/theme manually, moderate users and their roles, deploy a new WordPress website to a production server, etc...

WP-CLI will help you do all that in less time and automated as well, so It's a really a great tool that will help you react with your WordPress website.

Here's an example demonstrating how to run WP-CLI in a bash script to configure WordPress environment:

```
#!/bin/sh

# We will first check if the "/var/www/html" folder exist or not,
# if not we create it
if [ ! -d "/var/www/html" ]; then
    mkdir -p /var/www/html
fi

# We will cd into the folder
```

```
cd /var/www/html

# This downloads the WordPress core files, the option ( --allow-root ) will run
# and ( --version=5.8.1 ) specifies the version of WordPress that will get down
# and ( --locale=en_US ) sets the language of the installation to US English
wp core download --allow-root --version=5.8.1 --locale=en_US

# This will generate the WordPress configuration file, and the options ( --dbna
# are just placeholders that will get replaced once the script runs
wp config create --allow-root --dbname=${WP_NAME} --dbuser=${WP_ADMIN_USER} --d

# This will then install WordPress, and again, all the options are just placeho
wp core install --allow-root --url=${WP_URL} --title=${WP_TITLE} --admin_user=$

# This create a new WordPress user, and sets its role to author ( --role=author
wp user create "${WP_USER}" "${WP_EMAIL}" --user_pass="${WP_PASSWORD}" --role=a

# This is the command that will keep WordPress up and running
exec php-fpm7 -F -R
```

What is Redis Cache? And Why Do You Need it in WordPress?

Redis Cache, or Redis Object Cache, is an open-source, in-memory data structure store that can be used as database, cache, or message broker. It's a plugin for WordPress that improves the performance of your website by storing accessed data in memory, rather than querying the database each and every time that data is needed.

You need Redis Cache plugin in your WordPress website because it can improve the performance of the website and reduce the time that it takes to load all the data from your databases, which will cause in a better user experience, plus it will help your website rank higher in search engines, etc...

In this project of Inception, in order to Redis Cache to work perfectly, you need to add this line in your WordPress script:

- If you're using Alpine: chown -R nobody:nobody *
- If you're using Debian: chown -R www-data:www-data *

What does this basically do is that it changes the ownership of all files and

directories recursively (-R) to the user and group nobody:nobody or www-data:www-data

Plus, you also will need to add the `REDIS_HOST` as long as the `REDIS_PORT` to your WordPress `wp-config.php`

Here's and example:

```
sed -i "41 i define( 'WP_REDIS_HOST', 'redis' );\ndefine( 'WP_REDIS_PORT', '6379' );"
```

This will add 2 lines to the `wp-config.php` in the 41st line of the file, the first is `define('WP_REDIS_HOST', 'redis');` and the second is `define('WP_REDIS_PORT', '6379');`. Those 2 lines will specify the host and port that Redis Cache will use to establish the connection.

Adding them using `echo` seems to cause some problems, and Redis Cache does not want to establish the connection!

What is FTP? And How Does it Work?

FTP or File Transfer Protocol is a protocol that's used for transferring files between a client and a server over TCP/IP network, such as the internet. It provides a robust mechanism for users to upload, download, and manage files on remote servers.

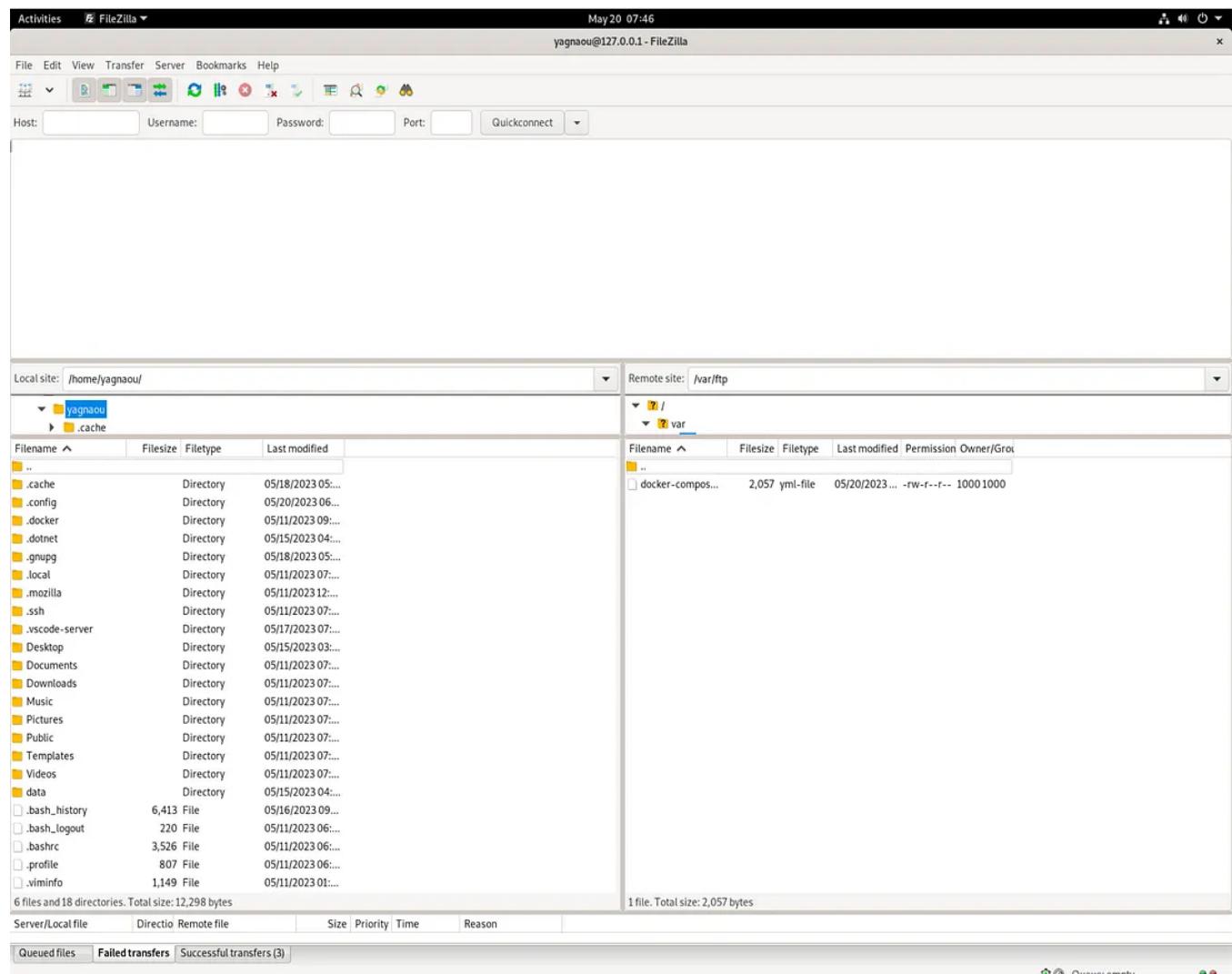
FTP works by opening two connections that link the 2 hosts (client and server) trying to communicate between each other, one connection is designed for the commands and replies that gets sent between the two clients, and the other connection is handles the transfer of the data.

How to Configure FTP? And How to Test If it's Working?

Before configuring and testing if FTP is working, you need to install FileZilla or other alternatives in your machine, which will allow you to transfer files between the two hosts easily and make sure that everything works as it should. I'll use FileZilla in my end

```
sudo apt-get install filezilla
```

Once installed, open it by typing the `filezilla` command in your terminal. A window like this will then popup



Now you have to configure the FTP so you can connect to the FileZilla, then, and only then you can start testing and transferring the data.

First, you will begin by installing the package needed for the FTP, and copying the config file to its proper directory. Here's the Dockerfile for the FTP

```
FROM alpine:3.14

# Installing the VSFTPD
RUN apk update && \
    apk add vsftpd

# Copying the config file to its proper directory
COPY ./conf/vsftpd.conf /etc/vsftpd/vsftpd.conf

# Here, you will create a new user, and give it the
# permissions needed so you can connect to FileZilla using it
RUN adduser -D -h /var/ftp USER_NAME && \
    echo "USER_NAME:USER_PASSWORD" | chpasswd && \
    mkdir -p /var/ftp && \
    chown -R USER_NAME:USER_NAME /var/ftp && \
    chmod 755 /var/ftp

CMD ["vsftpd", "/etc/vsftpd/vsftpd.conf"]
```

And here's the config file for the FTP

```
# This line specifies that the FTP server should listen for incoming connection
listen=YES

# This line disables anonymous access to the FTP server
# (ONLY authenticated users can get access)
anonymous_enable=NO

# This line enables local user access to the FTP server
local_enable=YES

# This line enables write access for authenticated users
# (authenticated users can upload/modify files on the FTP)
write_enable=YES

# This line tells the FTP to use the local system's time
local_umask=022

# This line enables the display of directory welcome messages
# (FTP will display a message when a user enter a directory)
```

```
dirmessage_enable=YES

# This line instructs the FTP server to use the local system's time settings
use_localtime=YES

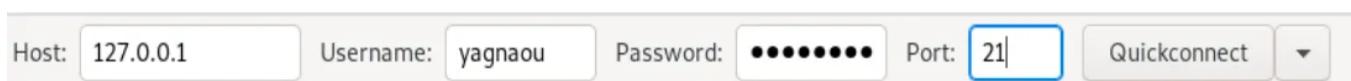
# This line enables logging of file transfer activities
# (FTP will create a log file containing infos about data transferring)
xferlog_enable=YES

# This line specifies whether the FTP server should use port 20 for active data
connect_from_port_20=NO

# This line disables the use of a seccomp sandbox
# (Seccomp is a security mechanism that restricts the system calls available to
seccomp_sandbox=NO
```

Now after setting everything up, open FileZilla, and connect using the following:

- Host: 127.0.0.1
- Username: USER_NAME
- Password: USER_PASSWORD
- Port: The one you specified in the Docker Compose, will usually be 21



If everything is done correctly, you will be able to connect to FileZilla without any error, and then you can start transferring data as you wish!

```
Status: Connecting to 127.0.0.1:21...
Status: Connection established, waiting for welcome message...
Status: Insecure server, it does not support FTP over TLS.
Status: Logged in
Status: Retrieving directory listing...
Status: Directory listing of "/var/ftp" successful
```

What is Adminer?

Adminer is a free open-source tool that allows you to easily view, edit, create, and

modify databases through a user-friendly interface. It supports a wide range of database systems, such as MariaDB, MySQL, PostgreSQL, SQLite, and many more... It is a single file application that doesn't require any installation, and that is what makes Adminer stand out and be preferred as a database manager among all the other alternatives.

I hope that this article helped you know just a little bit more about **Dockerization** and its related stuff as well ❤️

[Docker](#)[Docker Compose](#)[Dockerfiles](#)[Containers](#)[42 Network](#)[Follow](#)

Written by Youssef Agnaou

9 Followers · 0 Following

IT Engineering Student

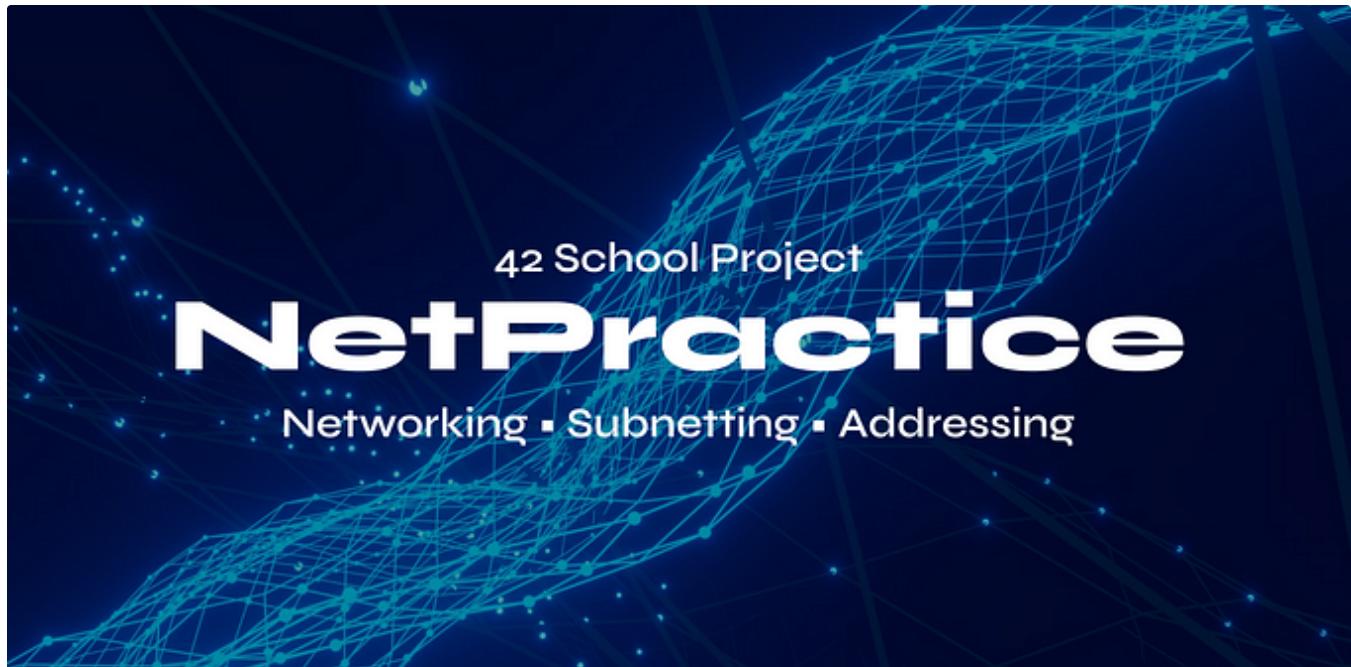
No responses yet



Write a response

What are your thoughts?

More from Youssef Agnaou



 Youssef Agnaou

NetPractice

42-School Project

May 21, 2023

See all from Youssef Agnaou

Recommended from Medium



In Javarevisited by Rasathurai Karan

Java's Funeral Has Been Announced....



Oh, Java is outdated! Java is too verbose! No one uses Java anymore!

3 Mar 7 1.4K 85



Always Free

24 GB RAM + 4 CPU + 200 GB

 @harendraverma2  @harendra21  @harendra21

 Harendra

How I Am Using a Lifetime 100% Free Server

Get a server with 24 GB RAM + 4 CPU + 200 GB Storage + Always Free

Spring Says Goodbye to @Autowired: Here's What to Use Instead

 Java Interview

Spring Says Goodbye to @Autowired: Here's What to Use Instead

Yes, starting with Spring Boot 3 and Spring Framework 6, Spring has been encouraging constructor-based dependency injection over field...

 Feb 21  394  18 

 Julius Hietala

Building a React Native Core ML Computer Vision App with Expo and YOLOv8

In this tutorial, we'll create an Expo module for image classification using YOLOv8 and build a simple app to showcase how a Core ML model...

Oct 2, 2024  11



Codebase	Size (LOC)	Current	Native	Speedup
VS Code	1,505,000	77.8s	7.5s	10.4x
Playwright	356,000	11.1s	1.1s	10.1x
TypeORM	270,000	17.5s	1.3s	13.5x
date-fns	104,000	6.5s	0.7s	9.5x
tRPC (server + client)	18,000	5.5s	0.6s	9.1x
rxjs (observable)	2,100	1.1s	0.1s	11.0x

 Andrew Zuo

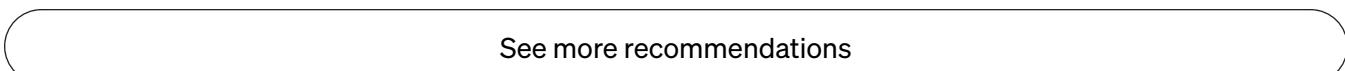
Why Microsoft Is Rewriting Their Typescript Compiler In Go

Microsoft has announced Typescript 7 (codenamed ‘Corsa’) despite them currently being on version 5.8.2. The reason why is that Typescript 7...

 5d ago  193  15 Dilshara Hetti Arachchige

Getting Started with Lynx: A Next-Gen Cross-Platform Framework

The world of cross-platform app development just got a major shake-up. ByteDance, the company behind TikTok, has introduced Lynx, a...

  231  6 See more recommendations