Web Server Project Report

Project Overview

For this project, I built a secure and manageable web server for a small family-owned business. The goal was to create something realistic, affordable, and secure without relying on expensive cloud infrastructure.

The main priorities were:

- Keep it secure using HTTPS and Cloudflare
- Make it easy to manage with Docker and YAML files
- Keep costs low
- Build a clean, professional-looking website

The site was deployed using a Cloudflare-managed domain and protected behind a tunnel so the real server IP is never exposed.

Environment & Tools Used

Operating System

I used **Ubuntu** as the base system. It is stable, widely supported, and works well for containerized services.

Docker

All services were deployed using **Docker**.
Instead of installing everything directly on the host, I ran applications inside containers. This keeps services isolated and easier to maintain.

I wrote YAML configuration files to define:

- Containers

- Networking rules
- Volume mappings
- Restart policies

This makes redeployment simple and repeatable.

Reverse Proxy

I deployed **Caddy** as a reverse proxy container.

Caddy automatically handles HTTPS certificates and routes traffic to the correct container internally. This removed the need for manual certificate setup and simplified routing.

Cloudflare

I used **Cloudflare** to:

- Manage DNS
- Hide the origin server IP
- Provide encrypted HTTPS traffic
- Add an extra security layer between users and the server

Instead of exposing ports directly to the internet, I set up a Cloudflare Tunnel. This allows traffic to reach the server securely without opening unnecessary ports.

Frontend

I built mock business websites using **Tailwind CSS** to demonstrate how the final site would look. The focus was on a clean, responsive layout that would be realistic for a small business.

How the Architecture Works

1. Ubuntu server runs Docker
2. Containers are defined using YAML

3. Caddy handles HTTPS and reverse proxy routing
4. Cloudflare Tunnel securely connects the public domain to the server
5. The origin IP is never directly exposed

All public traffic goes through Cloudflare first, then securely tunnels to Caddy, which routes it to the correct container.

This setup reduces the attack surface and follows basic Zero Trust ideas by not exposing internal services directly.

Security Decisions

- No direct public IP exposure
- Automatic HTTPS enabled by default
- Reverse proxy used instead of publishing multiple ports
- Container isolation for better service separation
- Minimal open ports on the host

The goal was to keep the setup simple but hardened enough to reflect how a small business could realistically deploy a secure site.

What I Learned

- How to deploy and manage Docker containers with YAML
- How reverse proxies actually route traffic internally
- How Cloudflare Tunnel protects origin infrastructure
- How HTTPS certificate automation works in practice

- How to design infrastructure that is secure but still easy to maintain

---

Conclusion

This project shows how a small business can run a secure and professional web server without expensive cloud services.

By combining Docker, Caddy, and Cloudflare, I built a setup that is:

- Secure
- Easy to manage
- Affordable
- Professional

It also gave me hands-on experience with Linux administration, container networking, reverse proxy architecture, and DNS-level security controls.