Practical Work 02 A

Introduction

This document details the steps taken to complete Practical Work 02 A as part of the Hedy - Calilegua DevOps Engineer course. The objective was to set up a fully functional WordPress installation on a Debian 12 virtual machine, meeting the following requirements:

- Use **Nginx** as the web server
- Use **PHP 8.3** with PHP-FPM as the PHP processor
- Use **MariaDB** as the database management system
- Create a new database and user for **WordPress**, not using the default credentials

The virtual machine was previously created in Practical Work 01 A using Vagrant. The goal of this practical work was to update the provisioning script to automate the installation and configuration of the required components.

In the following sections, the step-by-step process undertaken to achieve this objective will be detailed.

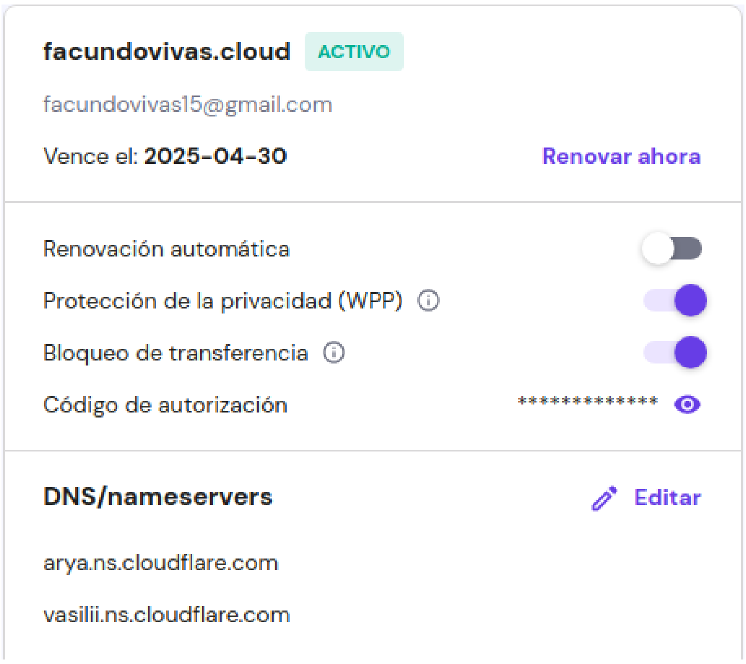
WordPress Deployment

Step 1: Securing a Domain Name

To fulfill the project requirements, securing a domain name from a reliable and cost-effective domain registrar was essential. After evaluating various options discussed in class, such as www.nic.ar or www.porkbun.com, Hostinger (www.hostinger.com) was chosen as the domain registrar for this project. The decision to go with Hostinger was based on recommendations from classmates and the affordable prices for domain registration.

With Hostinger as the domain registrar, facundovivas.cloud was secured to serve as the primary access point for the WordPress installation

Evidence:



facundovivas.cloud ACTIVO

facundovivas15@gmail.com

Vence el: **2025-04-30**

Renovar ahora

Renovación automática	off
Protección de la privacidad (WPP)	on
Bloqueo de transferencia	on
Código de autorización	*************

DNS/nameservers — Editar

arya.ns.cloudflare.com

vasilii.ns.cloudflare.com

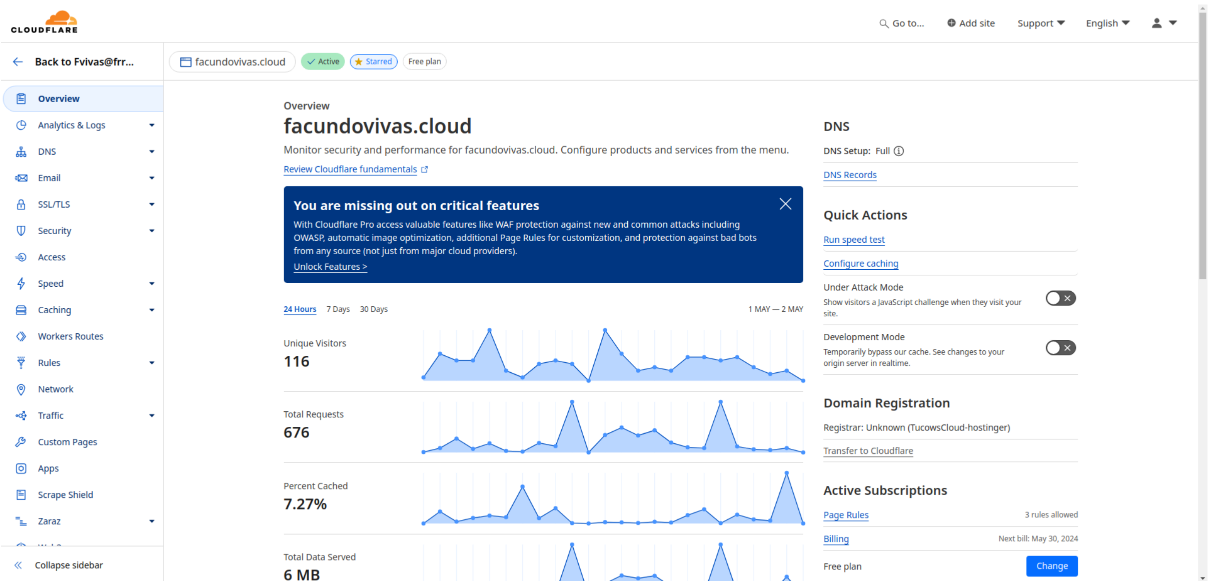
Step 2: Configuring Domain with Cloudflare DNS

After securing the facundovivas.cloud domain through Hostinger, the next step was to configure the domain's DNS (Domain Name System) settings. This step was crucial to ensure that the domain could properly resolve to the IP address of the Debian 12 virtual machine hosting the WordPress installation.

Adding a site to Cloudflare is a simple process that involves following the provided steps. The only action required from us as domain owners is to visit the domain registrar's website (in this case, Hostinger) and update the domain's nameservers to the ones provided by Cloudflare. Once the nameserver changes are made, it's just a matter of waiting for the DNS propagation and validation process to complete.

Troubleshooting: DNS Records
In my case, this step turned out to be quite frustrating. Initially, I assumed that the DNS configuration was done correctly and waited up to 48 hours for the DNS propagation to complete. However, it became clear that there was a configuration error: there was a failure during the step where Cloudflare analyzes the domain's DNS records. For some reason, Cloudflare was unable to recognize any of the existing DNS records, and it was prompting me to manually enter them.
After removing the domain from Cloudflare and attempting to reconfigure it, Cloudflare finally recognized the DNS records, and I was able to successfully activate the site.

Evidence:



Overview
facundovivas.cloud
Monitor security and performance for facundovivas.cloud. Configure products and services from the menu.
Review Cloudflare fundamentals

You are missing out on critical features
With Cloudflare Pro access valuable features like WAF protection against new and common attacks including OWASP, automatic image optimization, additional Page Rules for customization, and protection against bad bots from any source (not just from major cloud providers).
Unlock Features >

24 Hours 7 Days 30 Days — 1 MAY — 2 MAY

Unique Visitors 116
Total Requests 676
Percent Cached 7.27%
Total Data Served 6 MB

DNS
DNS Setup: Full
DNS Records

Quick Actions
Run speed test
Configure caching
Under Attack Mode — Show visitors a JavaScript challenge when they visit your site.
Development Mode — Temporarily bypass our cache. See changes to your origin server in realtime.

Domain Registration
Registrar: Unknown (TucowsCloud-hostinger)
Transfer to Cloudflare

Active Subscriptions
Page Rules — 3 rules allowed
Billing — Next bill: May 30, 2024
Free plan — Change

Step 3: Configuring Cloudflare Tunnel on the Virtual Machine

After activating the site on Cloudflare, the next step was to create a Cloudflare Tunnel towards the virtual machine. This tunnel would enable secure communication between Cloudflare's network and the WordPress installation running on the Debian 12 VM.

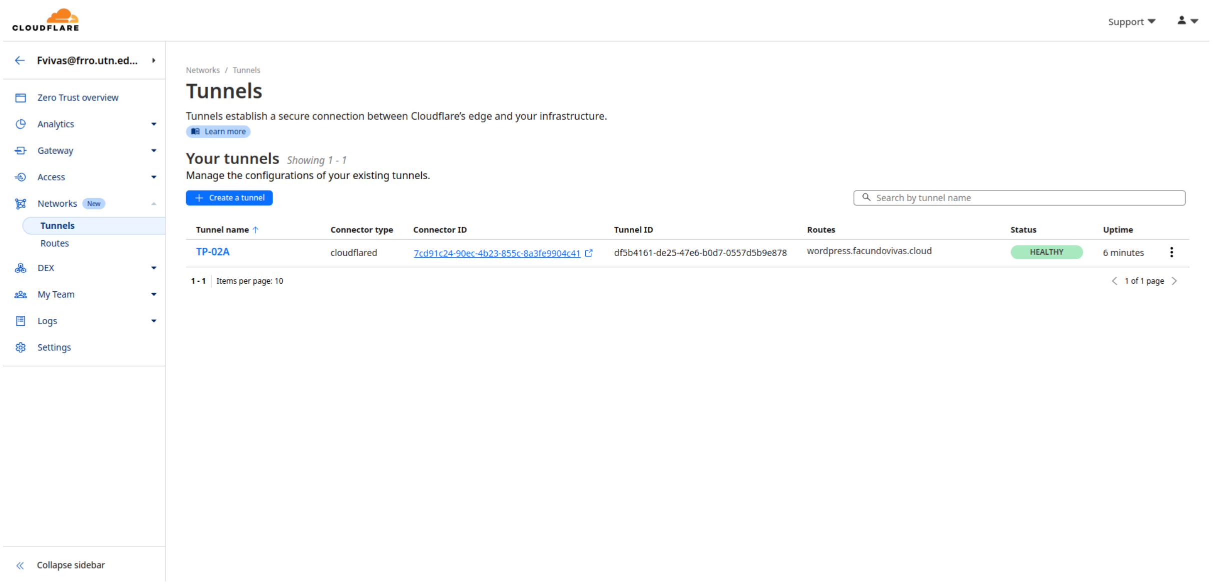
To set up the tunnel, the following steps were taken:

1. **Accessing Cloudflare Zero Trust**: In the Cloudflare dashboard, the "Zero Trust" functionality was accessed through the "Access" option.
2. **Creating a New Tunnel**: Within the new Zero Trust panel, the "Network" section was navigated to, and the "Tunnel" option was selected. Cloudflared

was chosen as the connector for establishing the tunnel.

3. **Updating the Vagrantfile**: At this point, the Vagrantfile required modification. The installation process was automated by creating a script called `cloudflred_tunnel.sh`. Cloudflare provided three commands that needed to be copied and pasted into the provisioning section of the Vagrantfile. These commands installed Cloudflared on the Debian 12 VM, which is the service responsible for managing the tunnel.
4. **Configuring Public Hostnames**: When the tunnel appeared as connected and healthy, the next step was to add the Public hostnames to the tunnel configuration, specifying the domain(s) that would be accessible through the tunnel (in this case wordpress.facundovivas.cloud).
5. **Saving the Tunnel Configuration**: After adding the public hostnames, the tunnel configuration was saved, completing the implementation of the Cloudflare Tunnel on the Debian 12 virtual machine.

Evidence:



Tunnels

Tunnels establish a secure connection between Cloudflare's edge and your infrastructure.

Your tunnels *Showing 1 - 1*

Manage the configurations of your existing tunnels.

Tunnel name	Connector type	Connector ID	Tunnel ID	Routes	Status	Uptime
TP-02A	cloudflared	7cd91c24-90ec-4b23-855c-8a3fe9904c41	df5b4161-de25-47e6-b0d7-0557d5b9e878	wordpress.facundovivas.cloud	HEALTHY	6 minutes

1 - 1 Items per page: 10 1 of 1 page

Step 4: Installing Required Services

To meet the project requirements, several services needed to be installed on the Debian 12 virtual machine. These services included Nginx as the web server, MariaDB as the database management system, PHP 8.3 with PHP-FPM for processing PHP code, and WordPress as the content management system.

The installation process was automated by creating a script called `wordpress_setup.sh`. This script contained the necessary commands to install and configure all the required components. The installation commands were detailed at the beginning of the script.

Step 5: Creating a custom MariaDB Database and User

One of the project requirements was to create a new database and user for WordPress, avoiding the use of default credentials. To meet this requirement, the appropiate command was added to the `wordpress_setup.sh`.

Instead of using the default root user, a new user called "*wordpress*" was created with the password "*YYFRYeKxS0r(kX^Ygj*". This user was granted full privileges over a newly created database named "*wordpress*".

Step 6: Configuring Nginx for WordPress

After installing the required services, the next step was to configure Nginx as the web server for the WordPress installation. This configuration was added to the `wordpress_setup.sh` script.

This configuration set up a new Nginx server block specifically for the WordPress site.

Here are the key aspects of the configuration:

1. **Listen on Port 80**: The server block was configured to listen on port 80 for incoming HTTP requests.
2. **Server Name**: The `server_name` directive was set to wordpress.facundovivas.cloud, which is the domain name acquired earlier.
3. **PHP Processing**: The `location ~ \\.php$` block included the necessary configuration for Nginx to pass PHP files to the PHP-FPM process for execution.

After creating the configuration file, it was symbolically linked to the `sites-enabled` directory in Nginx, and the Nginx service was ready to serve the WordPress site and handle incoming requests correctly.

Step 7: Installing and Configuring WordPress

After setting up the web server and database components, the next step was to install and configure the WordPress application itself. This process was also automated in the `wordpress_setup.sh`.

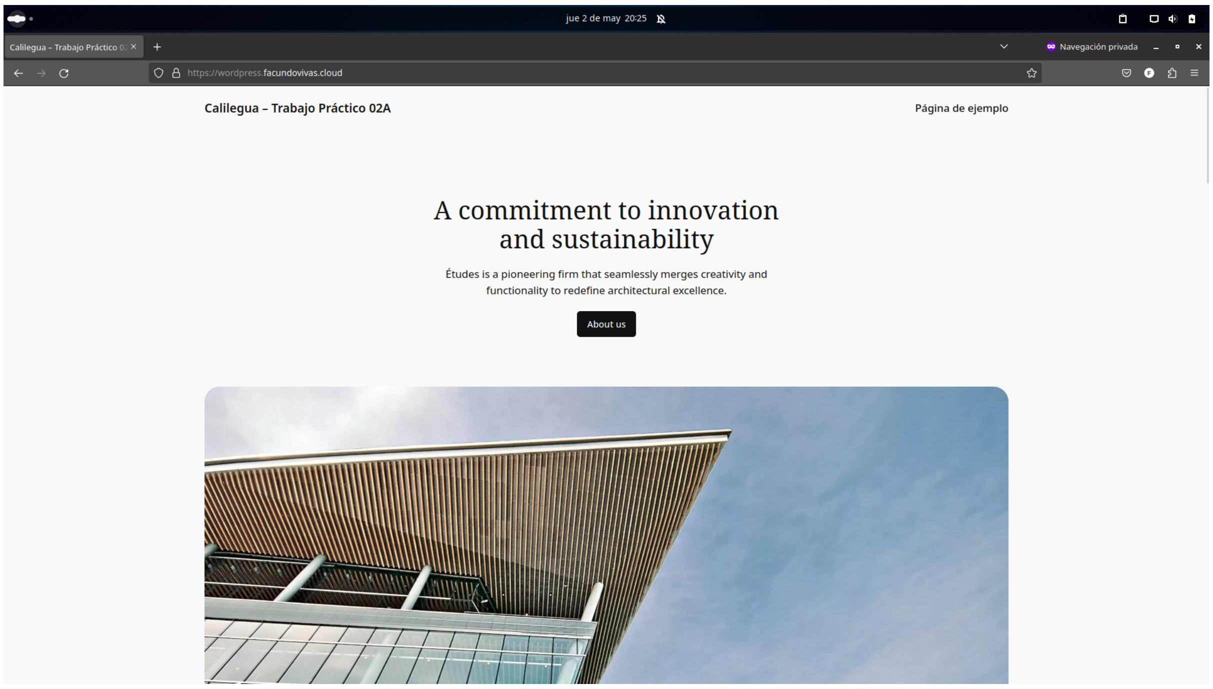
After executing this script, the WordPress installation was complete, and the site was accessible through the configured domain https://wordpress.facundovivas.cloud.

Step 8: Configuring WordPress via the Web Interface

After completing the installation and configuration steps outlined in the previous sections, the final step was to access the WordPress web interface and finalize the setup process.

By completing this final step of configuring WordPress via the web interface, the WordPress installation on the Debian 12 virtual machine was fully set up and ready for use, meeting the requirements of the assignment.

Evidence:


jue 2 de may 20:25
Calilegua – Trabajo Práctico 02A
Navegación privada
https://wordpress.facundovivas.cloud
Página de ejemplo
About us

Calilegua – Trabajo Práctico 02A

A commitment to innovation and sustainability

Études is a pioneering firm that seamlessly merges creativity and functionality to redefine architectural excellence.

References

The work was carried out following the steps outlined in the following guide, adapting it to the Debian 12 environment and the specific requirements of the practical work.

How to Install WordPress with LEMP on Ubuntu | DigitalOcean

WordPress has seen an incredible adoption rate among new and experienced engineers alike, and is a great choice for getting a website up and running efficien...

https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-with-lemp-on-ubuntu