

CapRover

Plataforma

Como

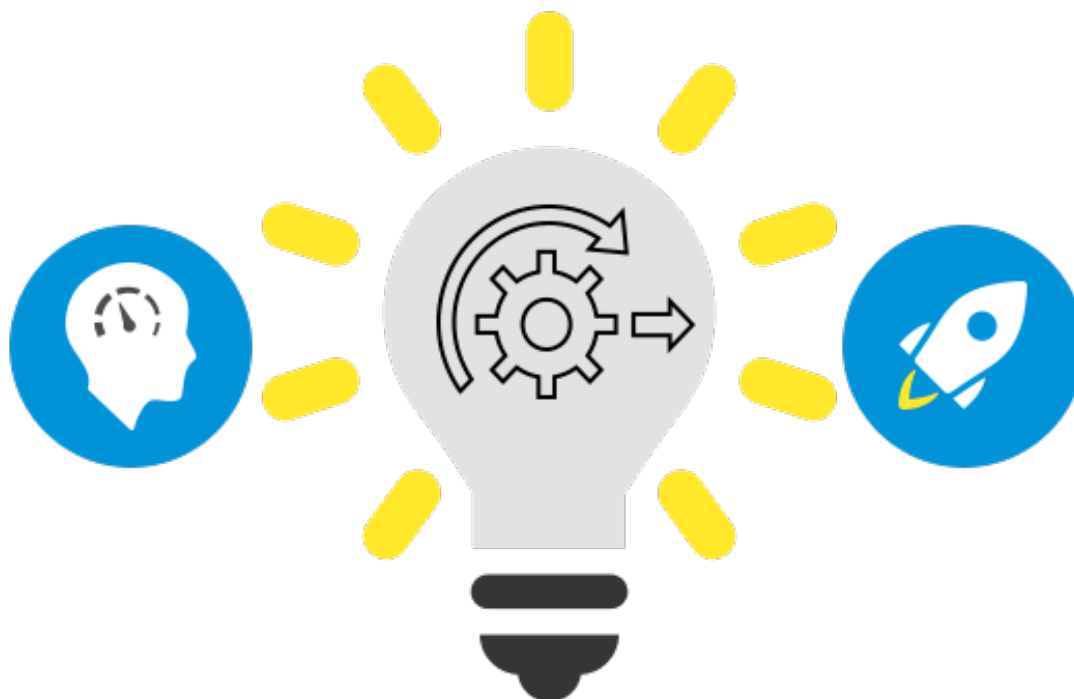
Servicio

PaaS

# Índice de contenido

|   |    |
|---|----|
| 1. OBJETIVOS DEL PROYECTO.....  | 1  |
| 1.1 Nacimiento de la Idea:.....   | 1  |
| 1.2 Objetivos a cumplir:.....   | 2  |
| 1.3 Los objetivos que pretendo cumplir:.....                                | 2  |
| 2. ESCENARIO.....   | 3  |
| 2.1 Explicación del escenario:.....   | 4  |
| 3. CONCEPTOS BÁSICOS.....   | 4  |
| 3.1. ¿Qué es la Computación en la Nube (Cloud Computing)?.....              | 5  |
| 3.2 ¿Qué es una PaaS?.....  | 6  |
| 3.2.1 ¿Como funcionan las PaaS?.....  | 6  |
| 3.2.2 Diferencia entre IaaS, PaaS y SaaS.....                               | 7  |
| 4. CAPROVER.....  | 8  |
| 4.1. ¿Qué es CapRover?.....   | 8  |
| 4.1.1. Versiones de CapRover.....   | 9  |
| 4.1.2. Términos y Condiciones.....  | 9  |
| 4.2. ¿Cómo funciona CapRover?.....  | 10 |
| 4.2.1 CLI e Interfaz Web.....   | 11 |
| 4.3 Métodos de Despliegue de Aplicaciones.....                              | 14 |
| 4.4. Archivo Captain-Definition.....  | 14 |
| 4.4.1. Ejemplos de Captain-Definition.....                                  | 14 |
| 4.4.2. Aplicación esta creada en NodeJS:.....                               | 15 |
| 4.4.3. Indicando directamente el contenido de un Dockerfile:.....           | 15 |
| 4.4.4. Mediante un archivo Dockerfile:.....                                 | 16 |
| 4.4.5. Mediante una imagen de Docker:.....                                  | 16 |
| 5. INSTALACIÓN.....   | 17 |
| 5.1.1 Pre-Requisitos.....   | 17 |
| 5.1.2 Requisitos del Sistema.....   | 18 |
| 5.2 Instalación en Debian 12 (VPS).....                                     | 19 |
| 5.3 Instalación en Debian 12 (Cliente).....                                 | 20 |
| 5.3.1 Instalación de la CLI de CapRover.....                                | 20 |
| 6. GESTIÓN DE APLICACIONES.....   | 24 |
| 6.1. Despliegue de aplicaciones.....  | 24 |
| 6.1.1 Despliegue de Book-Express (NodeJS).....                              | 24 |
| 6.1.2. Despliegue de NextCloud mediante “un clic”.....                      | 32 |
| 6.2. Despliegue Interconectando Aplicaciones.....                           | 49 |
| 6.2.1 Despliegue de Wordpress.....  | 50 |
| 6.3 Despliegue Mixto: Django.....   | 60 |
| 7. SOLUCIÓN DE ERRORES.....   | 70 |
| 7.1 Bucle infinito de creación de interfaces de red.....                    | 70 |
| 7.2 Error de eliminación de CapRover.....                                   | 71 |
| 7.3 Recuperar CapRover eliminado accidentalmente.....                       | 72 |
| 7.4 Recuperar la contraseña de la interfaz web.....                         | 73 |
| 8. VÍDEOS DE DEMOSTRACIÓN.....  | 74 |
| 8.1 Instalación de CapRover:.....   | 74 |
| 8.2 Demostración 1: Aplicación Biblioteca en NodeJS.....                    | 74 |
| 8.3 Demostración 2: Despliegue Automático de Nextcloud.....                 | 75 |
| 8.4 Demostración 3: Despliegue Interconectando Aplicaciones: Wordpress..... | 75 |
| 8.5 Demostración 4: Despliegue Mixto: Django.....                           | 75 |
| 9. CONCLUSIÓN / AGRADECIMIENTOS.....  | 75 |
| 10. BIBLIOGRAFÍA:.....  | 77 |

## 1. OBJETIVOS DEL PROYECTO.



### 1.1 Nacimiento de la Idea:

Como **comenté en el Pre-Proyecto**, este proyecto lo he elegido, porque **soluciona el problema** de la complejidad en la gestión de **desplegar las aplicaciones** necesarias para usuarios con menos experiencia en la administración de infraestructuras como por ejemplo programadores, o usuarios varios.

CapRover, una vez configurado, **despliega las aplicaciones**, junto con sus respectivos dominios, **directamente listos para usarse**.

**A nivel de empresa**, en una gran empresa con una **intranet / extranet** donde cada departamento necesite ciertas aplicaciones, pudiendo otorgar privilegios a cada departamento respectivamente para desplegar las aplicaciones que vayan necesitando.

Este trabajo **puede realizarse por un usuario/s que no tengan que estar directamente relacionados con la administración de sistemas**, en lo que a conocimientos se refiere, lo que permitiría una amplia libertad y rapidez, y sobretodo evitar los dolores de cabeza que puede suponer implementar una aplicación que encima tenga mala documentación, para alguien que no sea experto. Esto claro está, bajo la supervisión de alguien para posibles problemas, añadir nuevas aplicaciones, etc.

## **1.2 Objetivos a cumplir:**

Para la **defensa de este proyecto**, voy a cumplir una serie de objetivos, los cuales, una vez realizados todos, tendré **completamente operativa** una herramienta que me va a **permitir desplegar aplicaciones automáticamente**, con unos pocos clics, dependiendo del caso.

Con esto me refiero **desde aplicaciones conocidas** como por ejemplo **Wordpress**, a aplicaciones **menos conocidas** o de desarrollo personal, ya que esto **depende del lenguaje** en el cual se haya escrito la aplicación, y no en su funcionalidad. Si la aplicación funciona, **funcionará en CapRover** (Por norma general claro, hay excepciones), pero de todo esto **hablaré con más profundidad en su respectiva sección** más adelante, cuando profundice en CapRover.

## **1.3 Los objetivos que pretendo cumplir:**

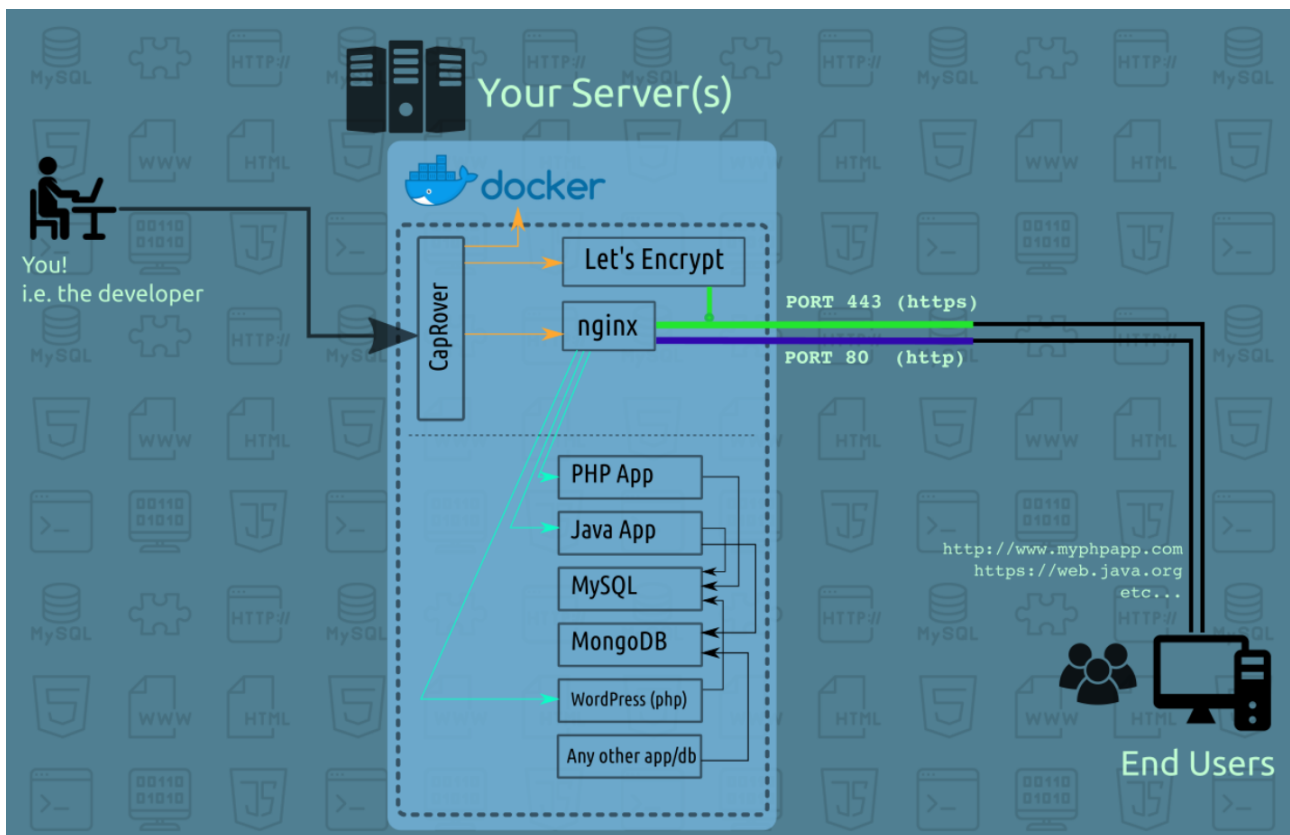
El objetivo principal, es el **despliegue de aplicaciones de forma simplificada y automática**, (dependiendo del caso), es decir, una **forma gratuita de tener servicios de pago** como Heroku, Microsoft Azure, etc para **desplegar las aplicaciones**, o en otras palabras, **una PaaS**.

Entre todas las funcionalidades que abarca Caprover, **una lista de los objetivos que serán cumplidos y demostrados** durante la presentación y defensa del proyecto son:

- **Despliegue Simplificado de Aplicaciones.**
- **Configuración de Dominios y Certificados SSL Automáticos.**
- **Gestión Centralizada y Accesible.**
- **Escalabilidad y Monitorización.**
- **Extensibilidad mediante Docker.**

- **Alta Disponibilidad y Resiliencia.**
- **Reducción de la Complejidad en Operaciones de DevOps.**
- **Costo-Eficiencia y Optimización de Recursos para Máquinas de bajo rendimiento.**
- **Y sobre todo, completa funcionalidad de las aplicaciones rápidamente.**

## 2. ESCENARIO.



El escenario para mi proyecto, va a limitarse **un pequeño escenario** en el cual se incluye, mi máquina **VPS**, y **mi portátil** como cliente remoto, ambas máquinas con **Debian 12 Bookworm**.



### 3.1. ¿Qué es la Computación en la Nube (Cloud Computing)?

Para poder explicar primero lo que es una PaaS o plataforma como servicio, primero hay que hablar de lo que es el denominado “**Coud Computing**” o **computación en la nube**.

Esta tecnología consiste en pocas palabras **en el uso de una red de servidores remotos**, los cuales se encuentran conectados a internet, **proporcionando uno o varios servicios remotos** como por ejemplo, **bases de datos, software, redes, administración de datos** etc.

Estos **servicios se almacenan en estos servidores**, los cuales **brindan este servicio** a otros equipos “clientes” sin la necesidad que dichos equipos clientes tengan la necesidad de tener instalado este tipo de servicios, **puediendo disfrutar de esto remotamente**, sin tener porqué cumplir los **altos requisitos** que conlleva estos servicios.

En pocas palabras, es una forma de **entrega de recursos informáticos a través de una red global**. Este tipo de **entrega de servicios** se dividen en distintos métodos de entrega, y uno de ellos precisamente es el que vengo a exponer en este proyecto: **Plataforma como Servicio (PaaS)**.

**Algunos de estos servicios son:**

- **Software como Servicio (SaaS).**
- **Plataforma como Servicio (PaaS).**
- **Infraestructura como Servicio (IaaS).**
- **Función como Servicio (FaaS).**
- **Red como Servicio (NaaS).**
- **Juegos como Servicio (GaaS).**

Los juegos como servicio lo incluyo en la lista, para que se haga notar que hoy en dia, casi todo forma ya del **Cloud Computing, incluido videojuegos**. Estos son solo unos ejemplos, de los muchísimos que existen.

Este tipo de “oferta de servicios” tiene muchas ventajas como por ejemplo que **un equipo de bajo rendimiento pueda disfrutar de servicios que requieran una gran demanda de recursos**, pero a

su vez, cuenta con una **desventaja fundamental** que es que **no se tiene control sobre el servicio contratado**, y que se **depende de una empresa externa**.

Para la mayoría de usuarios, **esto puede no ser ninguna desventaja o problema**, porque tienen a alguien encargado de reparar errores, implementar actualizaciones, dar mantenimiento etc. Pero para una **persona / empresa que trata con datos muy sensibles** sobre clientes o usuarios, que una empresa externa tenga el control sobre eso **no es lo más recomendable**, y más aún el no poder **gestionar uno mismo todo el servicio**.

Un ejemplo de esto, **es mi máquina VPS**. Mi máquina forma parte de una **Infraestructura como Servicio (IaaS)**, y mientras realizaba este proyecto, he tenido un problema que, como no tengo control total del despliegue de esta, **no puedo reparar por mi mismo**, y debo contactar con la empresa, y estar esperando a que me la solucionen, lo que puede llevar días, para un pequeño problema.

### 3.2 ¿Qué es una PaaS?

Una **Plataforma como servicio PaaS**, en pocas palabras es un **tipo de modelo de los servicios mencionados anteriormente**, que ofrece una **plataforma en la nube, para desarrollar, desplegar o gestionar aplicaciones**, sin los quebraderos de cabeza que esto supone, y sin tener que mantener el hardware, ya que de esto se encargan los proveedores de este servicio.

Como ya he mencionado, se trata ni mas ni menos de un **entorno integral en la nube**, que incluye todo lo que los desarrolladores o usuarios puedan necesitar para la gestión de sus aplicaciones.

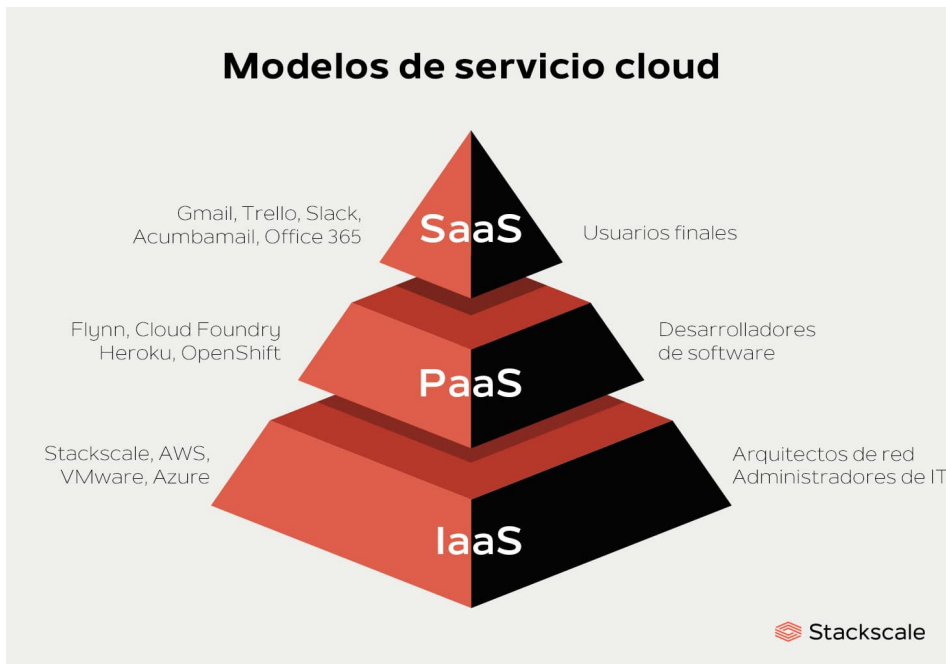
#### 3.2.1 ¿Como funcionan las PaaS?

En contraste con otros modelos de servicio como IaaS, o SaaS, esto esta **orientado al desarrollo de aplicaciones y software**. Dependiendo de la PaaS que se requiera, estas normalmente incluyen:

- **Infraestructura en la nube:** centros de datos, almacenamiento, equipo de red y servidores.
- **Software middleware:** sistemas operativos, frameworks, kits de desarrollo (SDKs), bibliotecas y más.
- **Interfaz de usuario:** interfaz gráfica de usuario (GUI), interfaz de línea de comandos (CLI), interfaz de API y, en algunos casos, las tres.



### 3.2.2 Diferencia entre IaaS, PaaS y SaaS.



Estos tres servicios, **son las categorías principales** digamos dentro del **Cloud Computing**, aunque como ya he mencionado más arriba hay muchas más categorías, y cada tipo de servicio ofrece una serie de recursos, **la diferencia fundamental entre ellos**, se trata de los **recursos a los que tenemos acceso para gestionar y los que no**. A continuación me refiero a “nosotros” como usuarios que disfrutamos de estos servicios. Por ejemplo:

**IaaS:** El proveedor gestiona la infraestructura (computación, almacenamiento, redes, virtualización), mientras que nosotros como usuarios gestionamos máquinas virtuales, sistemas operativos, aplicaciones y datos.

**PaaS:** El proveedor gestiona hardware y software para el desarrollo. En cambio, nosotros solo del código, las aplicaciones y los datos, sin preocuparte por la plataforma subyacente.

**SaaS:** El proveedor gestiona todo, desde la infraestructura hasta la aplicación, incluyendo actualizaciones y mantenimiento. Nosotros solo usamos la aplicación a través de Internet.

## 4. CAPROVER.



### 4.1. ¿Qué es CapRover?

Como explican en su [documentación](#), **CapRover es un gestor de implementación de aplicaciones** desde un servidor web, que permite **desplegar una gran variedad de aplicaciones** de distintos lenguajes como por ejemplo PHP, Python, NodeJS, etc de manera sencilla en pocos pasos, sin la necesidad de estar sufriendo los problemas que supone el despliegue de aplicaciones de forma manual, es decir, una plataforma como servicio PaaS.

Para poder comprender mejor lo que es, un buen ejemplo y con lo que **se suele comparar en muchas ocasiones, es con Heroku**, donde los usuarios podemos tener nuestras aplicaciones y utilizar Heroku para poder desplegarlas en internet de una manera relativamente sencilla.

Este servicio, aparte de ser **limitado**, hace un par de años **era gratis**. Pero actualmente, es un servicio de pago, lo pudimos comprobar el curso pasado cuando estaba en primero, donde debíamos desplegar una aplicación en Lenguajes de Marcas y Sistemas de Gestión de Información.

Pues **con CapRover, se elimina este problema**, siendo totalmente gratuito (aunque tiene versión pro), y completamente funcional. Y lo que es mejor, **despliega directamente las aplicaciones para que sean accesibles en internet directamente**, sin lios, ni procesos complicados, ni pagos.

Cabe destacar, que **CapRover cuenta con una variante de pago** mediante suscripciones. Este tema esta un poco oculto en su página web, pero para poder acceder a **“CapRover Pro”**, se hace desde [esta web](#).

#### 4.1.1. Versiones de CapRover.

##### CapRover Community Edition (CapRover gratuito):

- Versión de código abierto.
- Disponible de manera gratuita.
- Incluye todas las funcionalidades principales necesarias para desplegar y gestionar aplicaciones en servidores propios.

##### CapRover Pro (CapRover de pago):

- Notificaciones de estado de compilación.
- Mejoras de seguridad (2FA, alertas de inicio de sesión).
- Soporte técnico prioritario con SLA.

#### 4.1.2. Términos y Condiciones.

Otro punto a tener en cuenta son sus **términos y condiciones** que se encuentran [aquí](#) que se aceptan a la hora del despliegue del contenedor inicial / principal, y no se encuentran en su web oficial.

Un dato curioso sobre esto como he dicho, es que **no se especifica nada de ello en toda su página oficial, ni en la documentación**. Para poder encontrar estos hay que entrar a su [repositorio de GitHub oficial](#).

Por lo que se puede **leer en los términos**, legalmente hablando, **no puede considerarse un servicio**, ya que este, es **gratuito, opcional y de código abierto**.

Aunque buscando más información, es un tema de debate, porque por lo visto, han **hecho modificaciones** a la [licencia de Apache 2.0](#) que han generado discusiones porque con esto **no se cumpliría con las definiciones tradicionales de lo que representa el código abierto**, como la modificación y redistribución de ciertas características de pago de CapRover.

Por esto mismo, **se ha criticado a CapRover**, y algunos miembros de la comunidad sugieran que el proyecto se describa más apropiadamente como "**código fuente disponible**" en lugar de "**código abierto**". Todo esto se puede ir verificando directamente en su **repositorio oficial**, en el apartado de "**Issues**" y "**Discussions**".

Por otra parte, **CapRover recopila datos** como variables, imágenes, uso de aplicaciones, etc, que según sus términos y condiciones, **no se envían a terceros**. Esto solo se utiliza para "**poder medir los patrones de uso y optimizar la experiencia de los usuarios**".

**Esto puede desactivarse** declarando una variable env variable `CAPROVER-DISABLE-ANALYTICS=true` a CapRover.

Se puede **añadir esto a la línea de instalación o para un servicio ya existente**:

```
docker service update --env-add CAPROVER_DISABLE_ANALYTICS=true captain-captain
```

## 4.2. ¿Cómo funciona CapRover?

**CapRover funciona mediante contenedores Docker**, para ser más preciso, una vez que se despliega crea tres contenedores:

- **CapRover**: su propio contenedor principal que controla todo el despliegue maestro y resto de contenedores.
- **Nginx**: Para poder servir las aplicaciones directamente en producción, desplegándolas en internet, usarlo a modo de balanceador de carga como proxy inverso, etc.
- **Certbot**: Para gestionar los certificados de las aplicaciones, contenedores y sobre todo crear y gestionar los certificados para el despliegue seguro HTTPS de las aplicaciones.

Por otra parte, **se despliega un servicio por cada contenedor** de los anteriores, que vienen a ser instancias generadas por **Docker Swarm** (la herramienta de orquestación maestra / nativa de Docker para gestionar clústeres y coordinar contenedores), que controlan los contenedores. Hay que entender que **no es lo mismo un contenedor que un servicio**.

**Un contenedor**, en pocas palabras, es una **instancia basada en una imagen de Docker que ejecuta una determinada tarea**. En cambio **un servicio**, es una **forma de comunicación y control** que realiza Docker Swarm para poder **gestionar los contenedores**, su funcionamiento, réplicas, etc pudiendo gestionar múltiples contenedores dependiendo de su función.

Por ejemplo, para poder **listar los servicios** de mi máquina vps:

```
docker service ls
```

| <b>NAME</b>     | <b>MODE</b> | <b>REPLICAS</b> | <b>IMAGE</b>                      |
|-----------------|-------------|-----------------|-----------------------------------|
| captain-captain | replicated  | 1/1             | caprover/caprover:1.13.3          |
| captain-certbot | replicated  | 1/1             | caprover/certbot-sleeping:v2.11.0 |
| captain-nginx   | replicated  | 1/1             | nginx:1.27.2                      |

Lo que devuelve como he comentado **el servicio de cada respectivo contenedor**. Esto mismo se escala tanto a **redes, interfaces, volúmenes** etc.

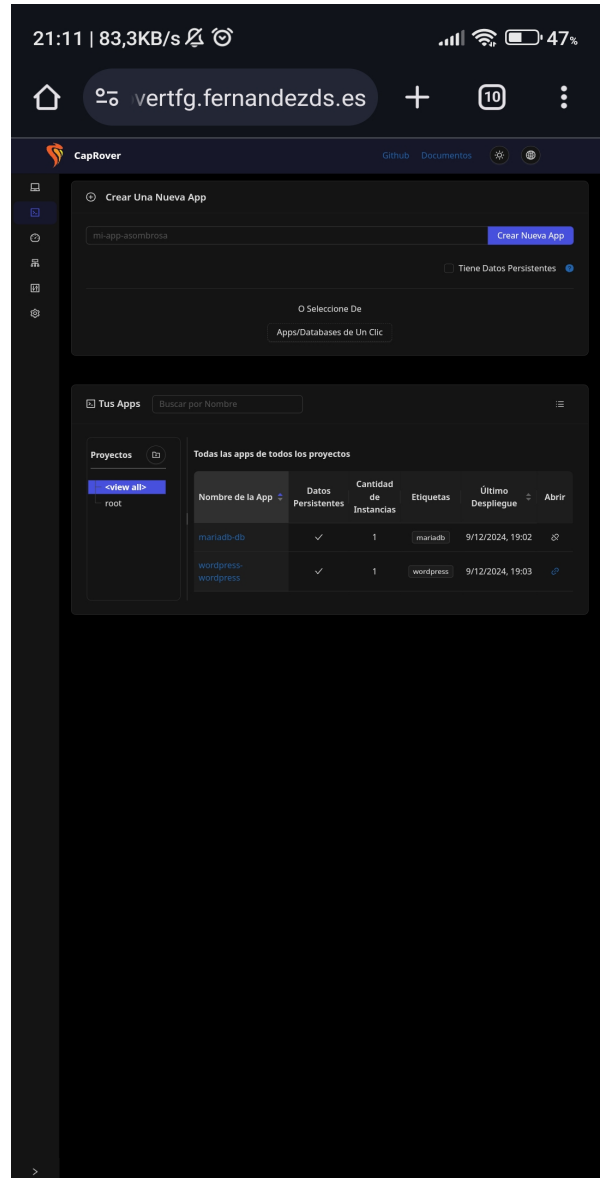
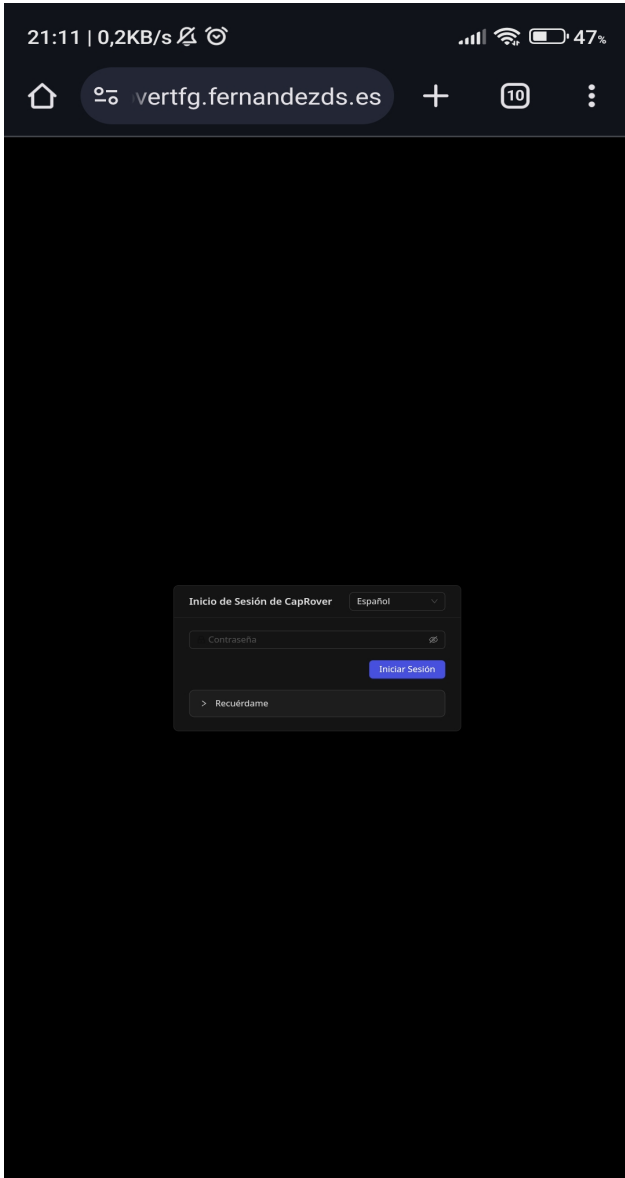
#### **4.2.1 CLI e Interfaz Web.**

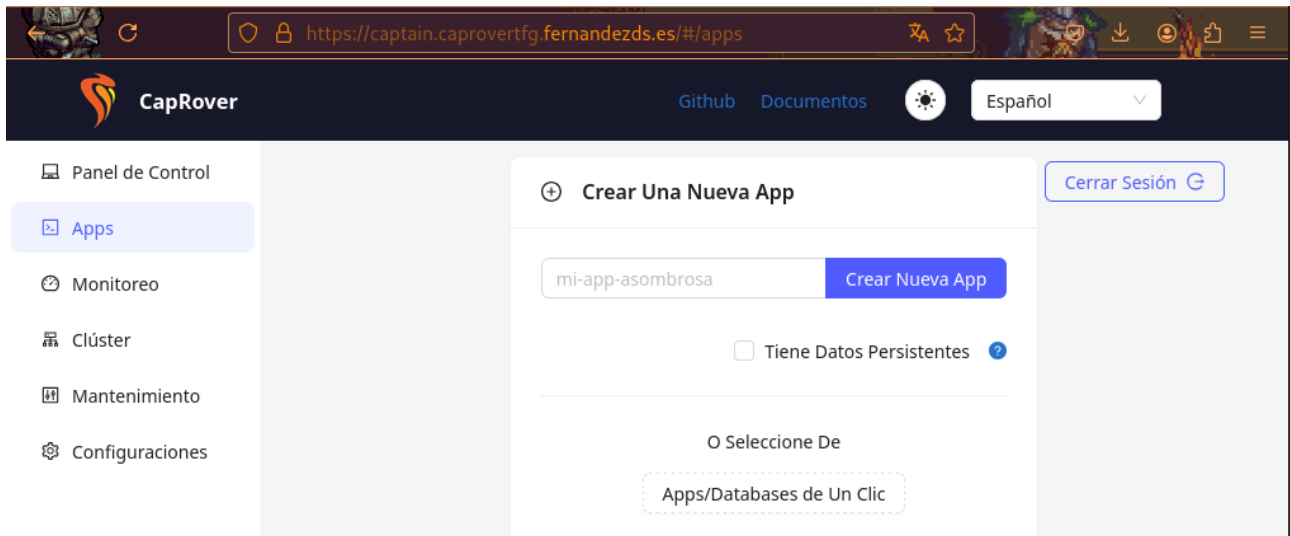
Como se muestra en su respectiva sección de instalación, **CapRover cuenta con tres formas principales de gestión:**

- **Interfaz Web:** la manera **más sencilla** y por lo que lo hace tan sencillo y poderoso a la vez. Una vez terminada la instalación, CapRover despliega la interfaz Web y proporciona un enlace, para que podamos crear y gestionar directamente las aplicaciones. Esto es **perfecto para usuarios que no tengan conocimientos avanzados** de uso de terminales o Docker.

Como la interfaz se encuentra en internet ya desplegada, **es accesible incluso desde dispositivos móviles, lo que amplía aún más su disponibilidad**. Como hablé al principio, que se pueda desplegar directamente aplicaciones completamente funcionales desde un teléfono por ejemplo, es una ventaja enorme, que puede solucionar **problemas de disponibilidad**.

Por ejemplo a continuación incluyo un par de capturas, para que se pueda comprobar que es **completamente funcional también desde un teléfono** (fue durante las demostraciones), y la siguiente imagen **de la propia interfaz web desde mi equipo Debian**.





- **CLI:** Viene a ser una **forma de despliegue de aplicaciones desde un cliente remoto** también, pero **con la diferencia que se usa por terminal, siendo perfecto para equipos sin interfaz gráfica**. Por ejemplo, si tengo una aplicación lista para desplegar en mi equipo configurado como cliente de CapRover, puedo desplegarla directamente con:

### ***caprover deploy***

Dependiendo del tipo de despliegue de la aplicación, puede variar, pero cuando termina el despliegue de nuestra aplicación en local de nuestro equipo cliente, la aplicación también se encuentra lista para usarse. De esto **hablo más en profundidad más adelante** en su respectiva sección.

- **Desde el propio servidor:** Desde la máquina VPS o servidor, es donde **tenemos el control completo de todo lo que ocurre en CapRover** ya que tenemos el control de Docker. Esto es imprescindible, para solucionar errores, y gestionar el servidor. Esto estaría más pensado para la persona encargada de la **gestión del servidor como el administrador**, más que a un usuario final.

### 4.3 Métodos de Despliegue de Aplicaciones.

Otra de las **grandes ventajas de CapRover**, es que, **permite el despliegue de aplicaciones de distintas formas**. No se limita solamente a un repositorio de una aplicación, o a subirla en el propio servidor, sino que dispone de más formas, para poder adaptarse al caso que queramos. Para ello primero debo hablar de un **archivo imprescindible para que funcionen** y carguen correctamente las aplicaciones, un archivo **JSON** llamado **captain-definition**.

### 4.4. Archivo Captain-Definition.

Este archivo es de **formato JSON**, que **debe estar presente obligatoriamente en cada aplicación** que quiera desplegar con CapRover. Este archivo lo que hace es **indicarle a CapRover como debe desplegar la aplicación**.

En este archivo es **donde se especifica, el método de despliegue** ya sea especificando los comandos del Dockerfile, o incluso si existe con el Dockerfile de la aplicación, o una imagen ya construida, etc. Este archivo **debe tener una ubicación específica** dentro de cada proyecto, por ejemplo:

- **Si la aplicación es NodeJS:** debe estar junto al archivo package.json
- **Si la aplicación es PHP:** debe estar junto al archivo index.php
- **Si en cambio es Python:** debe estar junto al archivo requirements.txt

Aunque no tiene que ser así en todos los casos, a continuación pondré algunos ejemplos mas usados según su documentación.

#### 4.4.1. Ejemplos de Captain-Definition.

El ejemplo más básico es para desplegar una aplicación, es **especificar el lenguaje** en el que esta escrita **y la versión** que esta utilizando. Por ejemplo:



#### 4.4.2. Aplicación esta creada en NodeJS:

```
{
  "schemaVersion": 2,
  "templateId": "node/8.7.0"
}
```

Esto suponiendo que la **versión de node** que este utilizando sea esa, para comprobar se con **node -v**. Esta versión si fuera diferente es la que hay que **sustituir en este archivo**. En uno de los ejemplos de despliegue hablo de esto también.

#### 4.4.3. Indicando directamente el contenido de un Dockerfile:

```
{
  "schemaVersion": 2,
  "dockerfileLines": [
    "FROM node:8.7.0-alpine",
    "RUN mkdir -p /usr/src/app",
    "WORKDIR /usr/src/app",
    "COPY ./package.json /usr/src/app/",
    "RUN npm install && npm cache clean --
force",
    "COPY ./ /usr/src/app",
    "ENV NODE_ENV production",
    "ENV PORT 80",
    "EXPOSE 80",
    "CMD [ \"npm\", \"start\" ]"
  ]
}
```

En este caso, es **indicarle al archivo captain-definition directamente las líneas que debe ejecutar** en vez de utilizar un Dockerfile como tal. Estas líneas son de muestra sacadas directamente de su documentación.

#### 4.4.4. Mediante un archivo Dockerfile:

Para **instalaciones más complejas**, se puede especificar directamente que **lea el archivo Dockerfile** de la aplicación en cuestión, una mejor forma si la aplicación necesita órdenes más complejas, o de tener una mejor organización. Por ejemplo:

```
{  
  "schemaVersion": 2,  
  "dockerfilePath": "./Dockerfile"  
}
```

En este caso, **se cambia el parámetro y se le indica la ubicación exacta del archivo Dockerfile**. En este ejemplo se encuentra en la misma ubicación que el archivo **captain-definition**.

#### 4.4.5. Mediante una imagen de Docker:

Otra forma es mediante el uso **directamente de una imagen de Docker**, para ello hay que indicar el nombre de la imagen de esta forma:

```
{  
  "schemaVersion": 2,  
  "imageName": "nginxdemos/hello"  
}
```

En este ejemplo se le indica la imagen mediante **imagenname**, y el **nombre de la imagen**. Estos los ejemplos más comunes de despliegues de aplicaciones personalizadas, en su documentación hay mas formas, pero **no me puedo detener a explicar cada parte** de todo lo que es capaz de hacer CapRover porque sino esta **memoria tendría una duración demasiado larga**.

## 5. INSTALACIÓN.

CapRover, es **compatible con sistemas Windows, como sistemas GNU/Linux**. En este proyecto he elegido principalmente mostrar el **despliegue en una máquina Debian (VPS)** que hará de servidor, y **otra máquina Debian**, que hará de cliente.

Por cliente no me refiero a un usuario final, sino a un **cliente a modo de gestión de los despliegues y manejo de CapRover**, sin la necesidad de estar en la máquina servidor principal.

### 5.1.1 Pre-Requisitos.

Antes de poder empezar con lo que sería la instalación, CapRover nos pide tener una **serie de prerequisites en la máquina**. Estos requisitos también se pueden aplicar a otro tipo de servicios y más si necesitan desplegar las aplicaciones en internet. Los “pre-requisitos” que necesitamos tener son:

- **Servidor público y FQDN:** Como ya vimos en clase, una forma de darle un nombre único a nuestra máquina.
- **Dominio:** Como vamos a desplegar las aplicaciones directamente operativas en internet, necesitamos tener previamente configurado un dominio, junto con su respectivo registro DNS.
- **Registro A DNS:** Este paso podría ser opcional, pero como vamos a desplegar una cantidad de aplicaciones, lo recomendable es crear un registro tipo A \*.caproverfg.fernandezds.es, en mi caso. De esta forma se garantiza que cualquier nuevo despliegue esté automáticamente en producción directamente en internet.
- **Docker:** Como CapRover trabaja con contenedores Docker, es indispensable tenerlo previamente instalado.
- **Puertos:** Según la documentación de CapRover, necesitamos tener abiertos los puertos 80,443, 3000, 996, 7946, 4789, 2377 de TCP, y por otro lado 7946, 4789, 2377 de UDP.

### 5.1.2 Requisitos del Sistema.

Según su documentación, los **requisitos mínimos de funcionamiento** de una máquina para que pueda ejecutar correctamente CapRover son:

#### Arquitectura de CPU:

- Compatible con cualquier arquitectura de CPU.
- Soporte para AMD64 (X86), ARM64 y ARMV7.

#### Sistema operativo recomendado:

- Verificado en **Ubuntu 20.04** y Docker 19.03.
- **Ubuntu 22.04** es funcional y recomendado, con soporte hasta abril de 2027.
- Se desaconseja Ubuntu 18.04 (fin de soporte)

#### RAM mínima:

- Requiere al menos **1 GB de RAM** (512 MB puede ser insuficiente para el proceso de compilación).

#### Entorno de Docker:

- Compatible con configuraciones basadas en Docker según la documentación oficial.

Para este caso **ha sido ejecutado en Debian 12 Bookworm**, y es totalmente funcional, por lo que en otras distribuciones similares debería ser compatible perfectamente. Tengo que aclarar que como **solo dispongo de 1GB de RAM** (que luego es menos dependiendo de los servicios que tengo), da **muchos problemas por falta de recursos, pero al menos, funciona.**

Para lograr que funcione he tenido que **dejar la máquina en un estado de funcionamiento mínimo** deteniendo unidades y servicios no relacionadas con esto, porque me daba cuenta, que con los despliegues la máquina acaba petando y provocando errores extraños, ya que **hablaré de ellos al final del documento, en la parte de solución de errores.**

## 5.2 Instalación en Debian 12 (VPS).

Una vez con los requisitos cumplidos, **CapRover servidor se instala de forma sencilla**, mediante el uso de **un contenedor** a modo de nodo maestro en nuestro servidor. Este contenedor será el **encargado principal de gestionar y desplegar distintos contenedores** para las aplicaciones que vayamos desplegando.

Para **comenzar la instalación** ejecuto en mi VPS:

```
docker run -p 80:80 -p 443:443 -p 3000:3000 -e ACCEPTED_TERMS=true  
-v /var/run/docker.sock:/var/run/docker.sock -v /captain:/captain  
caprover/caprover
```

Este comando lo que hace es **desplegar un contenedor principal**, contando con los **puertos 80:80, 443:443 y 3000:3000** para que puedan ser **accesibles externamente**.

Por otro parte establece una variable de entorno en el propio contenedor para poder **aceptar los términos y condiciones de CapRover**, ya que sino, no funciona.

También **monta el directorio captain en el host** justo en la misma ruta, para archivos persistentes de uso de CapRover, como configuraciones o certificados, etc.

Y por último se **especifica la imagen de Docker** que se va a utilizar para el contenedor.

Ejecutado este comando **empezará a descargar la imagen de Docker**, como cualquier otro contenedor. Una vez se haya completado, para que este totalmente desplegado hay que darle unos segundos.

Ahora **pasamos ahora al equipo que va a gestionar** todo el despliegue y gestión de CapRover y sus aplicaciones. En mi caso, **mi Debian 12 en mi portátil**.

### 5.3 Instalación en Debian 12 (Cliente).

Como he mencionado anteriormente, cuando hablo de cliente, **no me refiero a un cliente o usuario final**, sino a una **máquina distinta a la máquina servidor**, desde la cual se va a gestionar y configurar todos los parámetros de CapRover, junto con los despliegues de las aplicaciones, directamente **sin tener que interactuar directamente con el servidor VPS**, aunque se puede hacer **también desde cualquier equipo, gracias a su interfaz web**, desplegada directamente en internet.

Todo esto **será posible gracias a la CLI** de CapRover.

#### 5.3.1 Instalación de la CLI de CapRover.

Para poder instalar la CLI de CapRover, voy a necesitar **previamente tener tener instalado npm** (Node Package Manager), que viene a ser un **gestor de paquetes para Node.js**, que es una plataforma de desarrollo para poder ejecutar JavaScript. Esto me va a permitir instalar bibliotecas y herramientas de Node.js, justo lo que necesito en este caso. Así que para ello primero de todo lo instalo, como he dicho en la máquina Debian de mi portátil:

```
sudo apt install npm nodejs
```

Esto puede tardar unos minutos en terminar la instalación. Una vez terminada para poder ver la versión que tengo instalada:

```
node -v
```

```
npm -v
```

**Nota:** La versión de node, será importante más adelante durante una de las demostraciones.

Ahora que **npm esta instalado**, ya puedo **empezar la instalación de la CLI** de CapRover de la siguiente manera:

```
npm install -g caprover
```

*Andrés Fernández de Santaella*

Donde **-g** indica que el **paquete estará disponible** en cualquier parte del sistema, y no solo para un único proyecto, es decir, que este paquete podrá ser ejecutado desde cualquier ubicación de mi sistema, independiente del directorio de proyecto que este.

Una vez instalado, ahora es tan sencillo como **ejecutar el propio “asistente” de instalación**, donde me va a preguntar una **serie de parámetros de mi máquina VPS**, donde se encuentra alojado mi CapRover principal. Para empezar, ejecuto el comando:

***caprover serversetup***

Lo que **ejecutará el “asistente”** de la instalación:

**Setup CapRover machine on your server...**

**have you already started CapRover container on your server? Yes**

**IP address of your server: 82.165.14.4**

**CapRover server root domain: caprover.fernandezds.es**

**new CapRover password (min 8 characters): [hidden]**

**enter new CapRover password again: [hidden]**

**"valid" email address to get certificate and enable HTTPS:  
andresfernandezds@gmail.com**

**CapRover machine name, with whom the login credentials are stored locally:  
captain-01**

**CapRover server setup completed: it is available as captain-01 at  
https://captain.caprover.fernandezds.es**

**For more details and docs see CapRover.com**

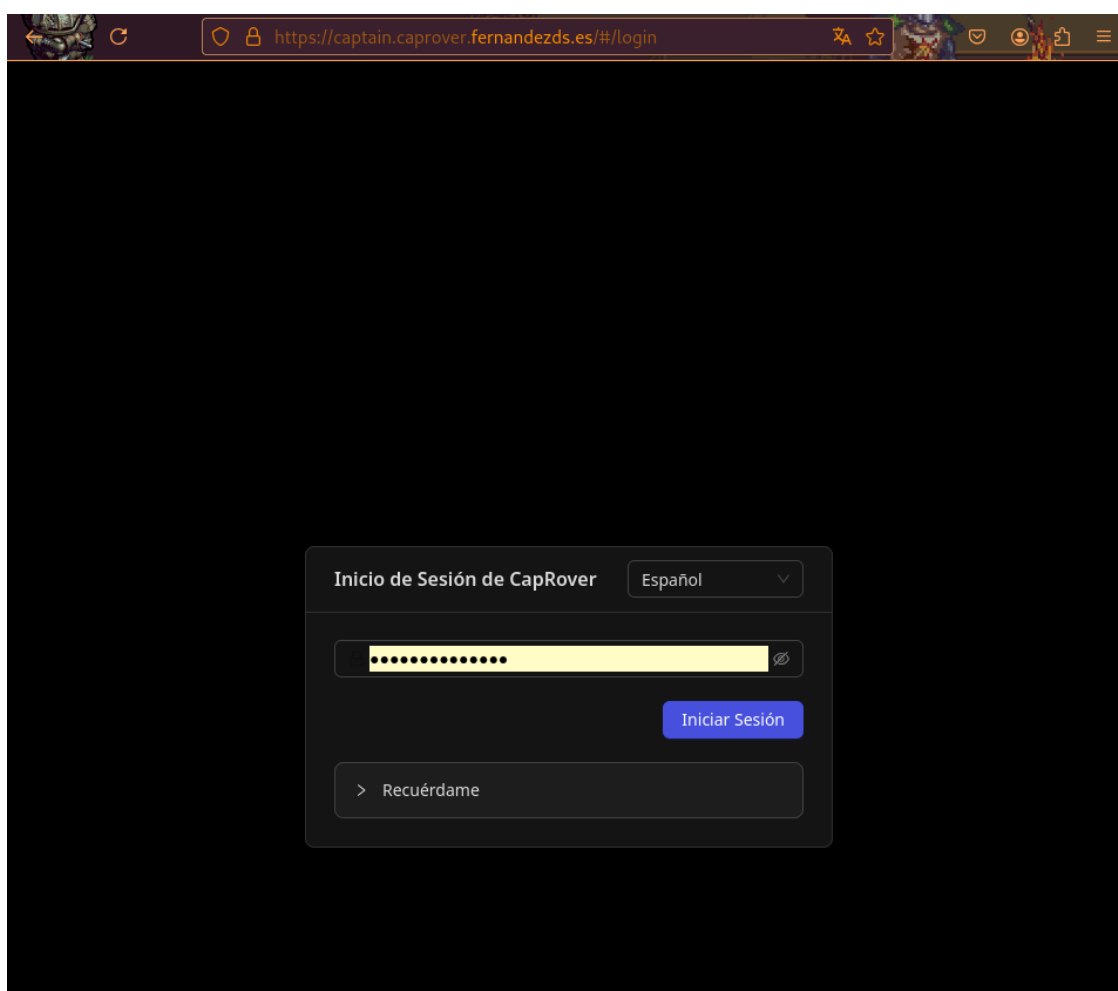
Ahora como he dicho antes, **nos va a preguntar una serie de parámetros del VPS**, como:

- Si el contenedor en la VPS esta listo para usarse.
- **La dirección IP pública** de mi servidor.

Andrés Fernández de Santaella

- **Dominio Raíz:** caprover.fernandezds.es (Para que este paso funcione, he tenido que haber creado el registro A que mencioné en los pre-requisitos.)
- **Una contraseña para CapRover**, que necesitaré para iniciar sesión en la CLI.
- **Una dirección de correo válida** para los certificados HTTPS.
- **Un nombre para la máquina**, en este caso la dejo por defecto.

Una vez terminado, **nos facilita una dirección HTTPS**, para que podamos **acceder para empezar a gestionar CapRover**. En este caso <https://captain.caprover.fernandezds.es> (obsoleto)

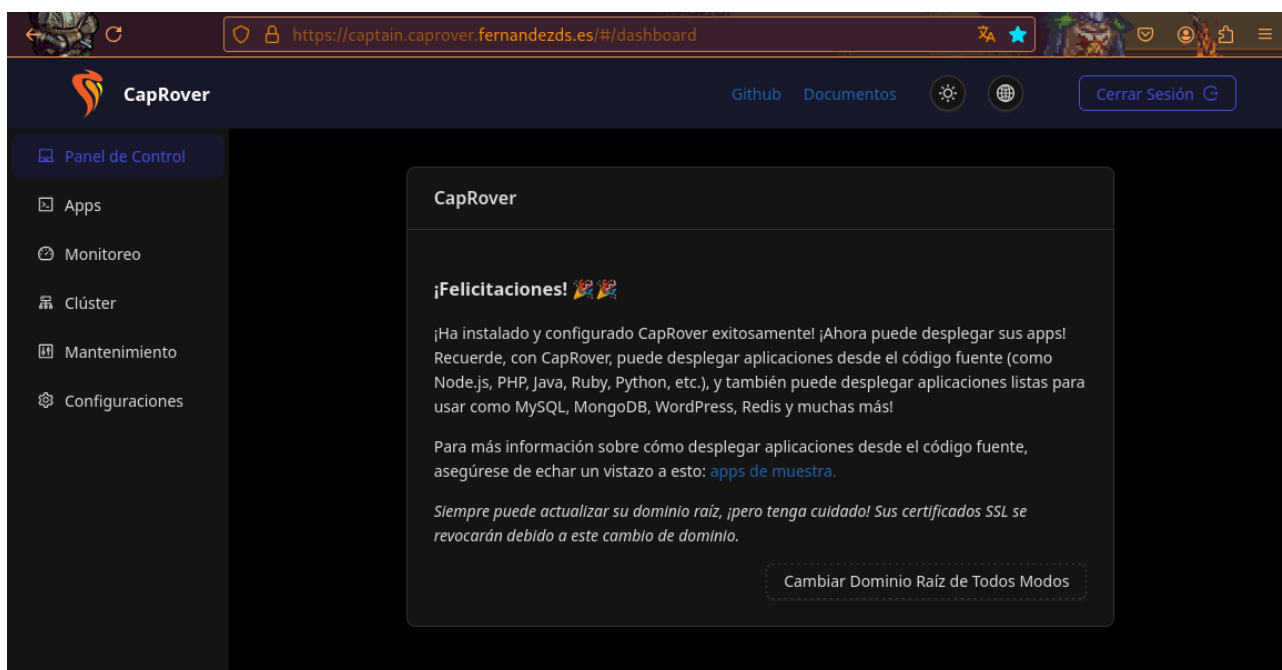


Al entrar en la dirección proporcionada por CapRover, que se encuentra **alojada en el VPS**, nos pide la contraseña que he introducido durante el proceso de instalación de antes. Una vez dentro ya



Andrés Fernández de Santaella

me encuentro ante la **CLI de CapRover totalmente funcional y lista**, para poder empezar a configurar las aplicaciones que vaya a implementar.



**Nota:** el dominio actual **pasa de caprover.fernandezds.es a caprovertfg.fernandezds.es** por problemas durante los despliegues, tuve que desplegarlo todo de nuevo en varias ocasiones. Esto solo afecta a las capturas hechas previamente, a los enlaces, pero por lo demás es exactamente igual.

## 6. GESTIÓN DE APLICACIONES

Como he mencionado en su sección, **CapRover permite el despliegue de aplicaciones de distintas formas**. No cubriré un despliegue de cada forma porque sino seria demasiado, pero si haré un despliegue de las formas más comunes.

### 6.1. Despliegue de aplicaciones

Para los despliegues de las aplicaciones **he elegido cuatro aplicaciones**, de las más comunes que son **WordPress, Nextcloud, Django, y Book-Express**. Durante la demostración y defensa del proyecto, si sobra tiempo puedo hacer el despliegue de alguna aplicación más si fuera posible.

#### 6.1.1 Despliegue de Book-Express (NodeJS).

**Book-Express**, es un ejemplo de una aplicación muy simple para almacenar libros, algo parecido a la que vimos en clase, pero más sencilla, al menos bajo mi opinión. Es un buen ejemplo para empezar con los despliegues de las aplicaciones.

Para **empezar con la demostración** parto de la base de tener **ya descargada** en mi equipo los archivos de la aplicación. Para que se ejecute correctamente, hay que recordar primero que **hay que configurar correctamente el archivo captain-definition**. Y para ello primero, como mencioné que seria importante mas adelante comprobar que **versión de node tengo instalada**, hay que ajustar el fichero para adecuarlo a mi versión actual quedando así:

```
{  
  "schemaVersion": 2,  
  "templateId": "node/18.19.0"  
}
```

Al ser una aplicación simple, no necesita de configuraciones extras en este fichero.

En la imagen superior se ve la **estructura de la aplicación**, junto con la **versión de node**, y el contenido del fichero **captain-definition**.

```
ls
```

```
captain-definition  Dockerfile  docs  node_modules  package.json  
package-lock.json  README.md  src
```

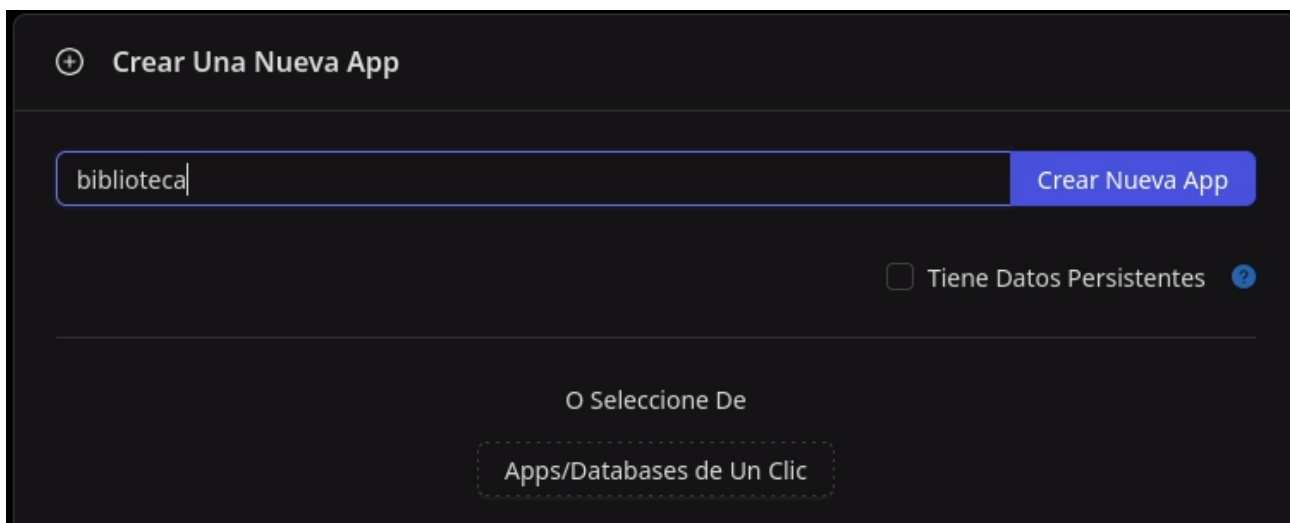
```
node -v
```

```
v18.19.0
```

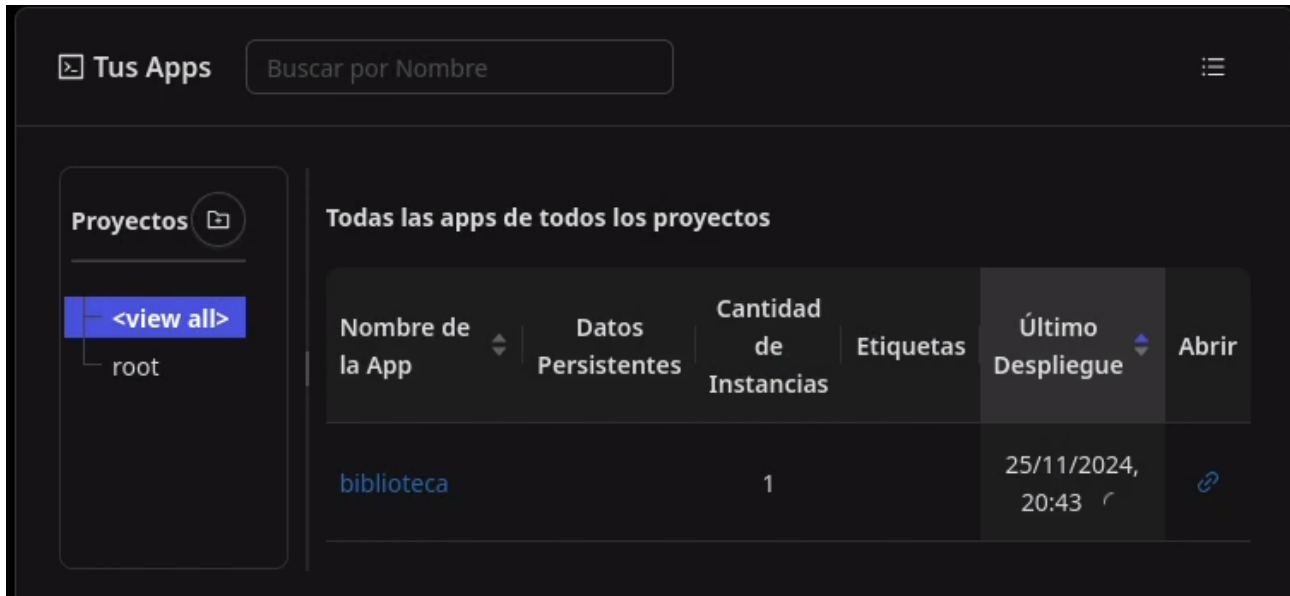
```
cat captain-definition
```

```
{  
  "schemaVersion": 2,  
  "templateId": "node/18.19.0"  
}
```

Antes de realizar el despliegue, **hay que crear un “proyecto”** digamos, para la aplicación, al ser una aplicación manual, creo primero donde se va a ejecutar.



Esto **crea en el servidor el proyecto (espacio)** para que este disponible en el **momento del despliegue** que veremos más adelante.



El siguiente paso es ejecutar el despliegue, con `capover deploy`. Al realizar esto, CapRover nos hará unas preguntas:

- **Máquina:** Nos pide que elijamos la máquina en la que tenemos el servidor (por si hubiera más de una, en mi caso solo una `captain-01`.)
- **Aplicación:** En este caso, en la lista aparece el proyecto o “espacio” que acabo de crear en la interfaz web.
- **Rama:** como las aplicaciones en este caso es de un repositorio, pregunta que rama quiero desplegar. En este caso es la master.
- **Archivos:** Y por último nos indica que los archivos que no han sido declarados con un commit o los del archivo `gitignore` no se incluirán en el despliegue y que si quiero continuar.

`capover deploy`

Preparing deployment to CapRover...

\*\*\*\* Protip \*\*\*\*

You seem to have deployed `biblioteca` from this directory in the past, use `--default` flag to avoid having to re-enter the information.

? select the CapRover machine name you want to deploy to: `captain-01`

Andrés Fernández de Santaella

Ensuring authentication...

? select the app name you want to deploy to: biblioteca

? git branch name to be deployed: master

? note that uncommitted and gitignored files (if any) will not be pushed to server! Are you sure you want to deploy? Yes

...

Successfully built 2c8af363d823

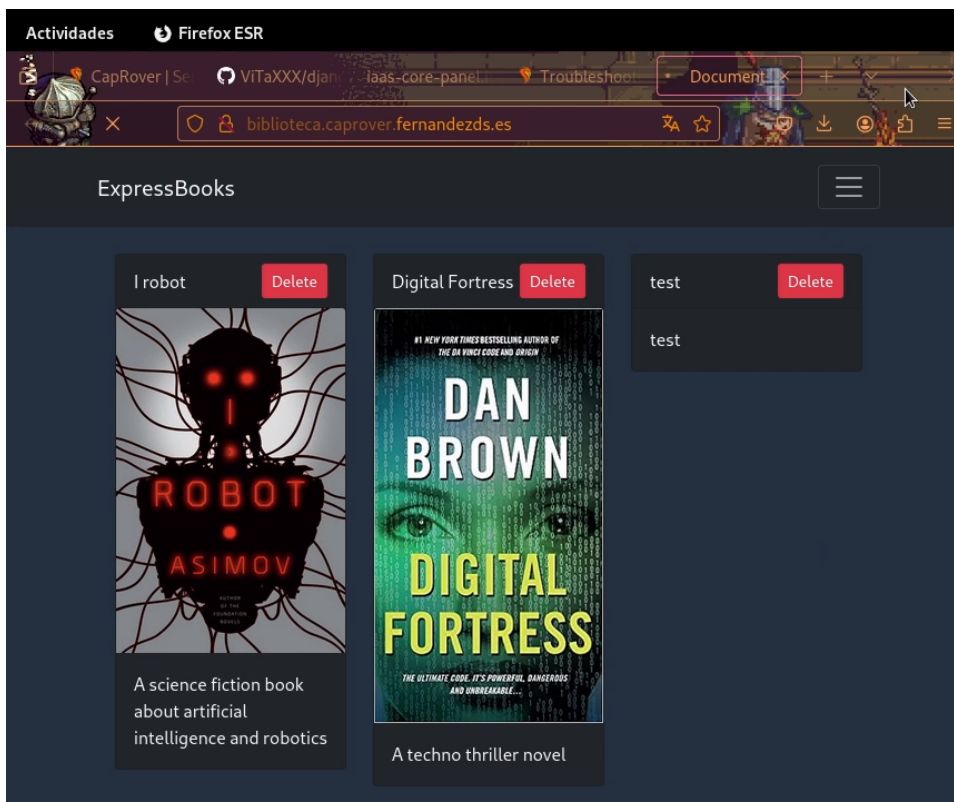
Successfully tagged img-captain-biblioteca:latest

Build has finished successfully!

Deployed successfully biblioteca

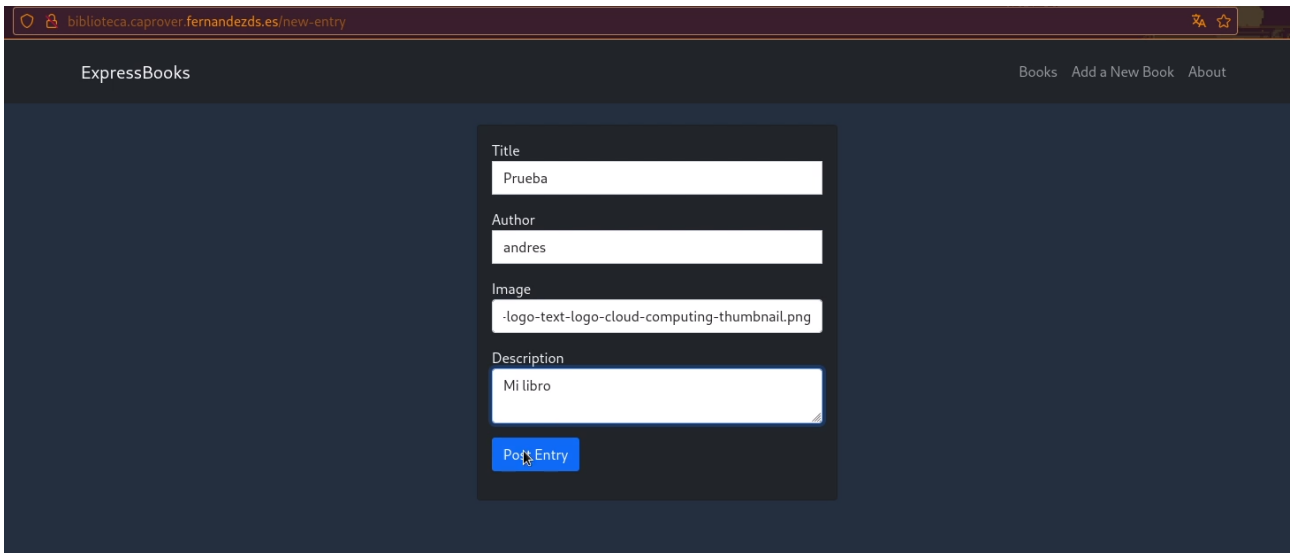
App is available at <http://biblioteca.capover.fernandezds.es>

Una vez **terminado el proceso de despliegue**, CapRover automáticamente la **despliega y la sirve directamente en internet**. Si accedo al enlace que ha proporcionado, y si todo ha ido bien, debe mostrarse la aplicación. En este caso ha **funcionado correctamente**, aunque aún **no esta desplegada como HTTPS**.

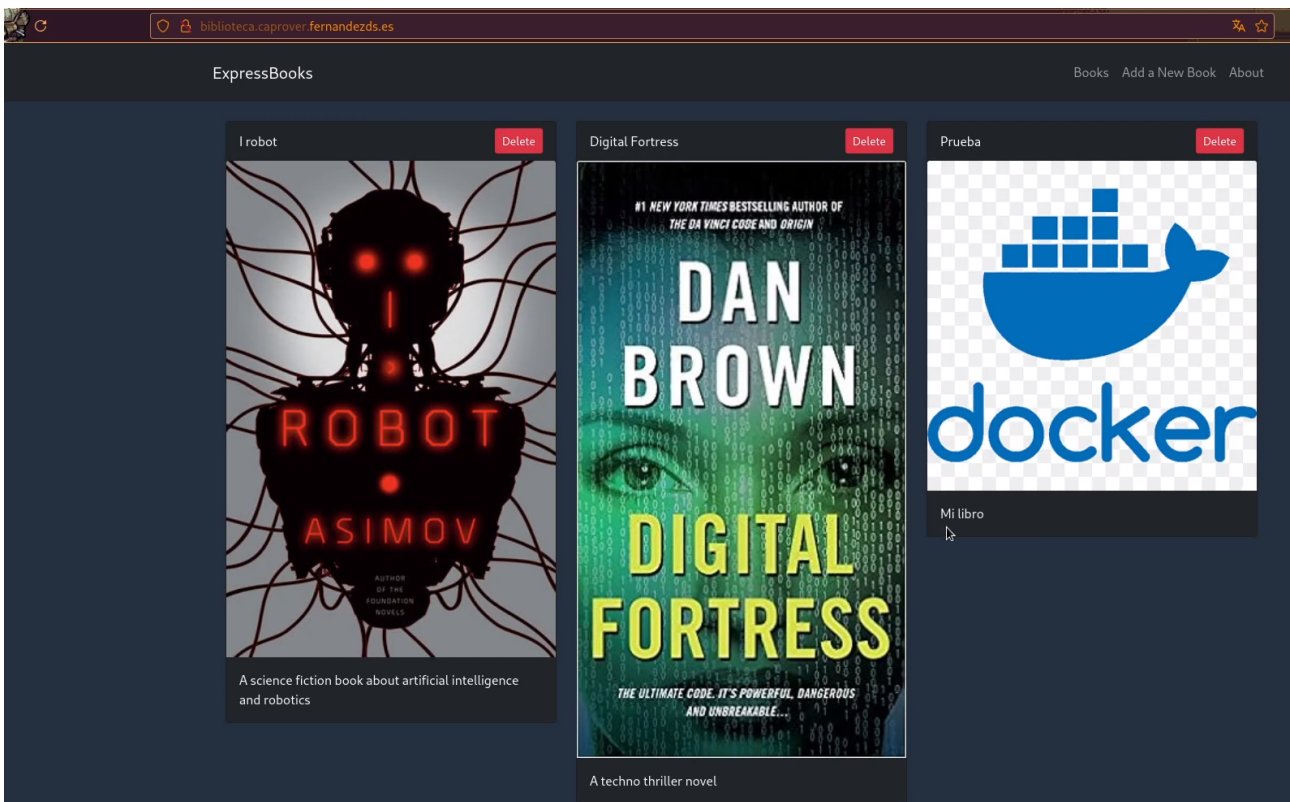


Andrés Fernández de Santaella

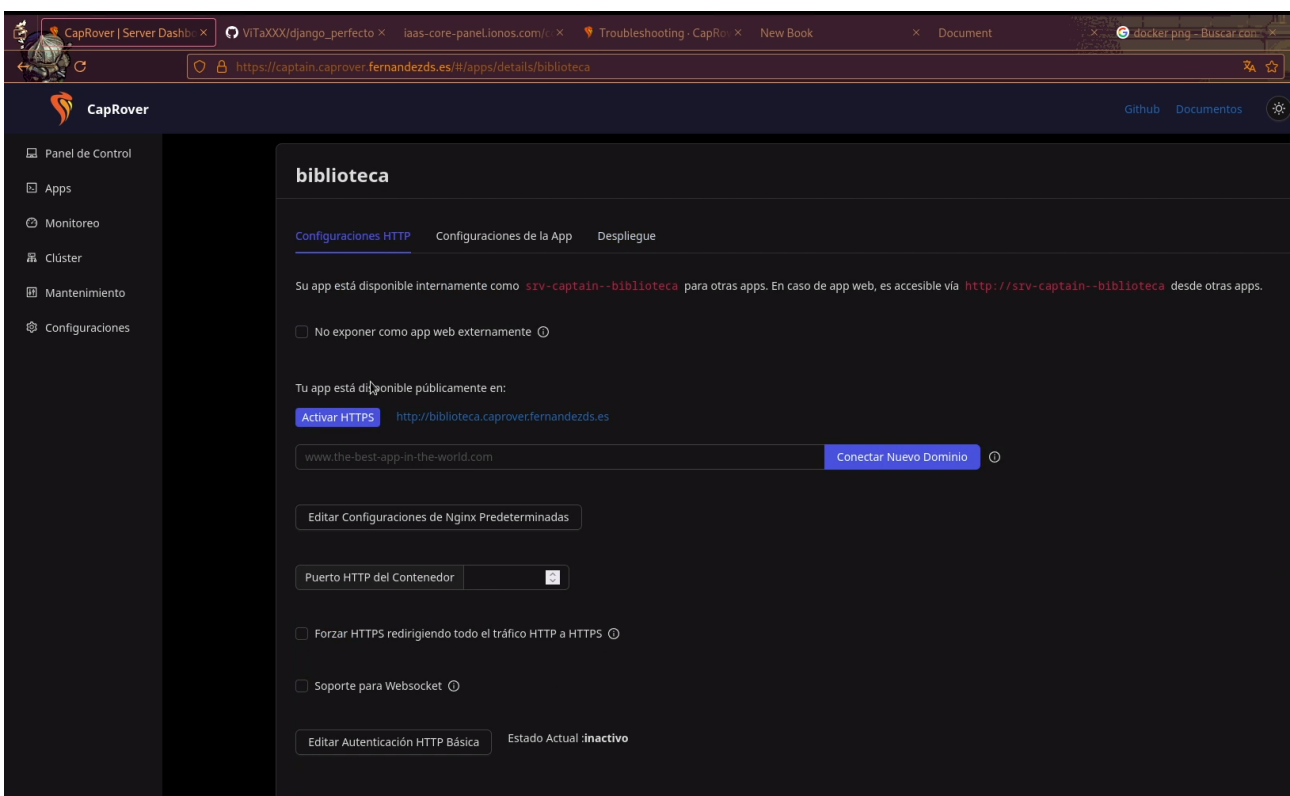
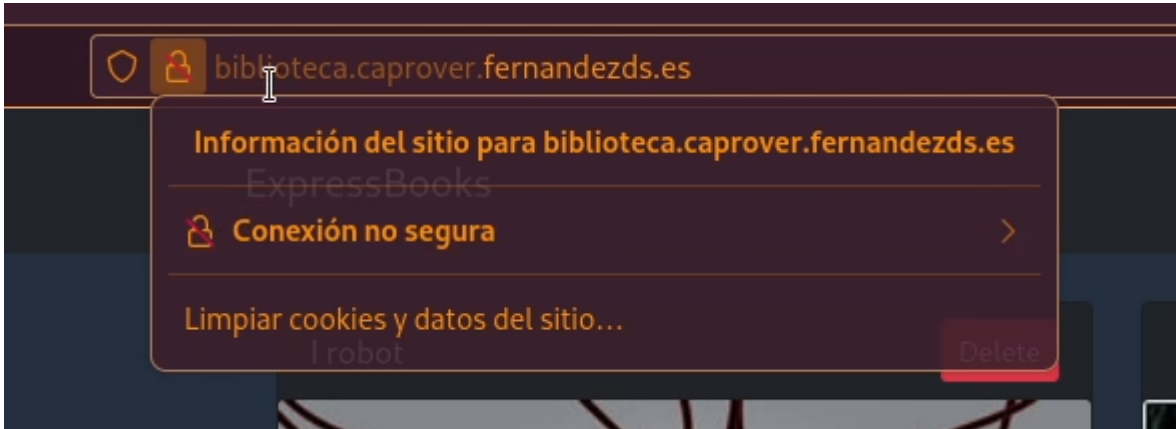
Para comprobar que funciona perfectamente voy a proceder a **añadir un libro de prueba** (en la opción add a new book)



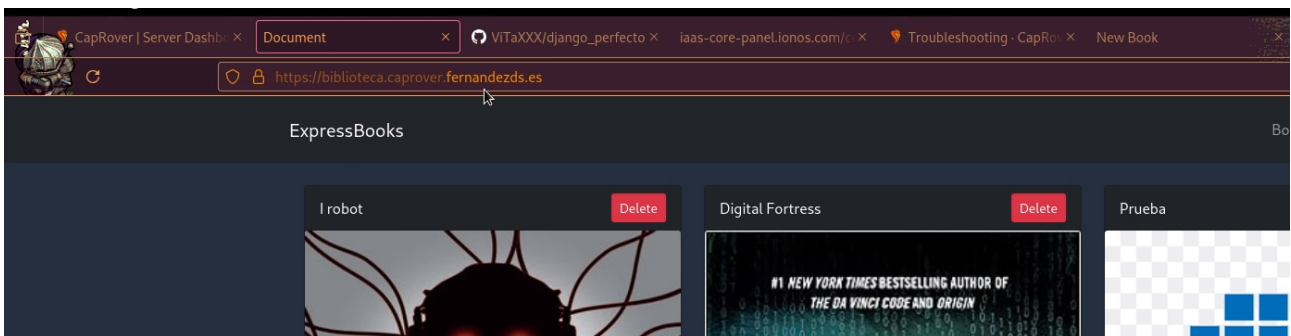
Y como se ve, se **añade correctamente al resto y sigue funcionando** en internet totalmente funcional y desplegada.



**Para terminar** con esta aplicación simple, voy a proceder a **activar el modo HTTPS**, que con CapRover, una vez configurado, **es tan simple como darle a un botón**:

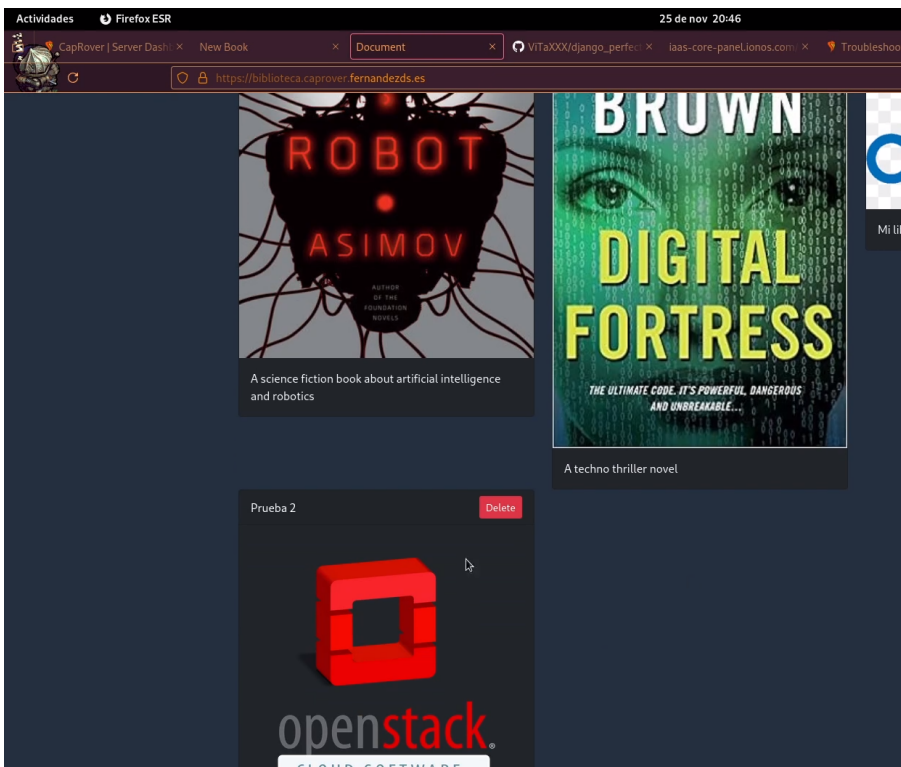
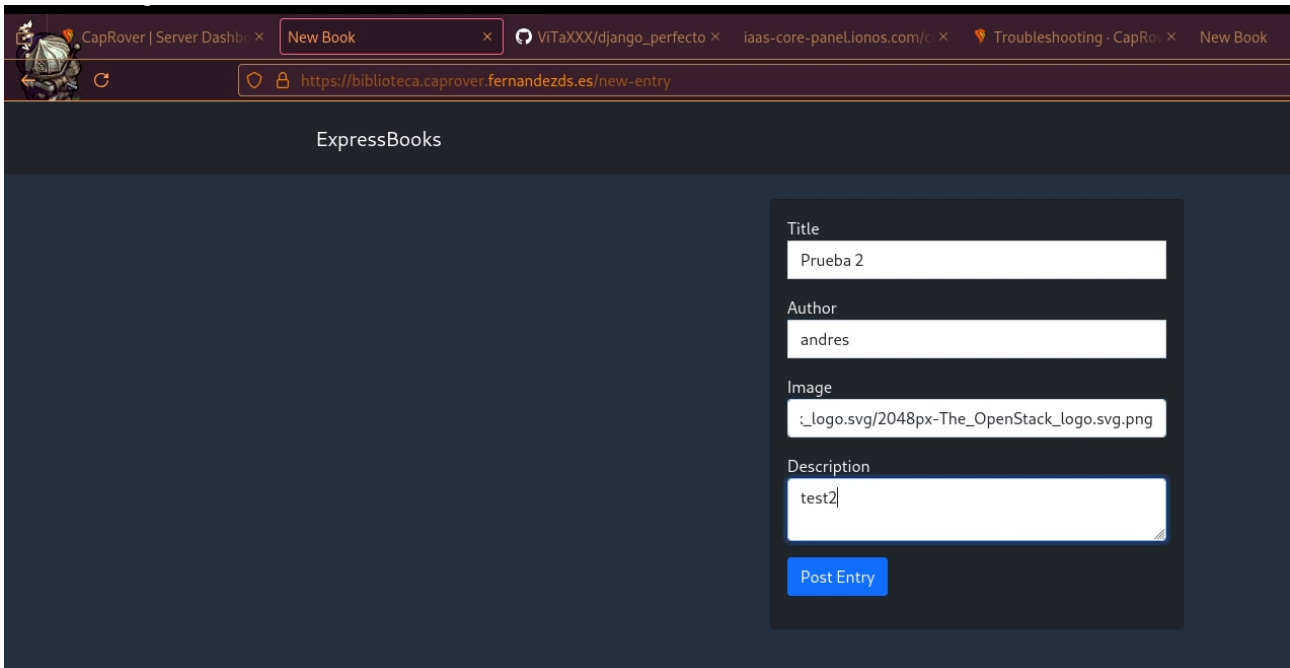


Tardará unos segundos, cuando termina, el dominio **pasa a estar ya configurado** totalmente para usar **HTTPS**.



Por último, para comprobar que **sigue siendo totalmente funcional**, voy a introducir un ultimo ejemplo a la aplicación.





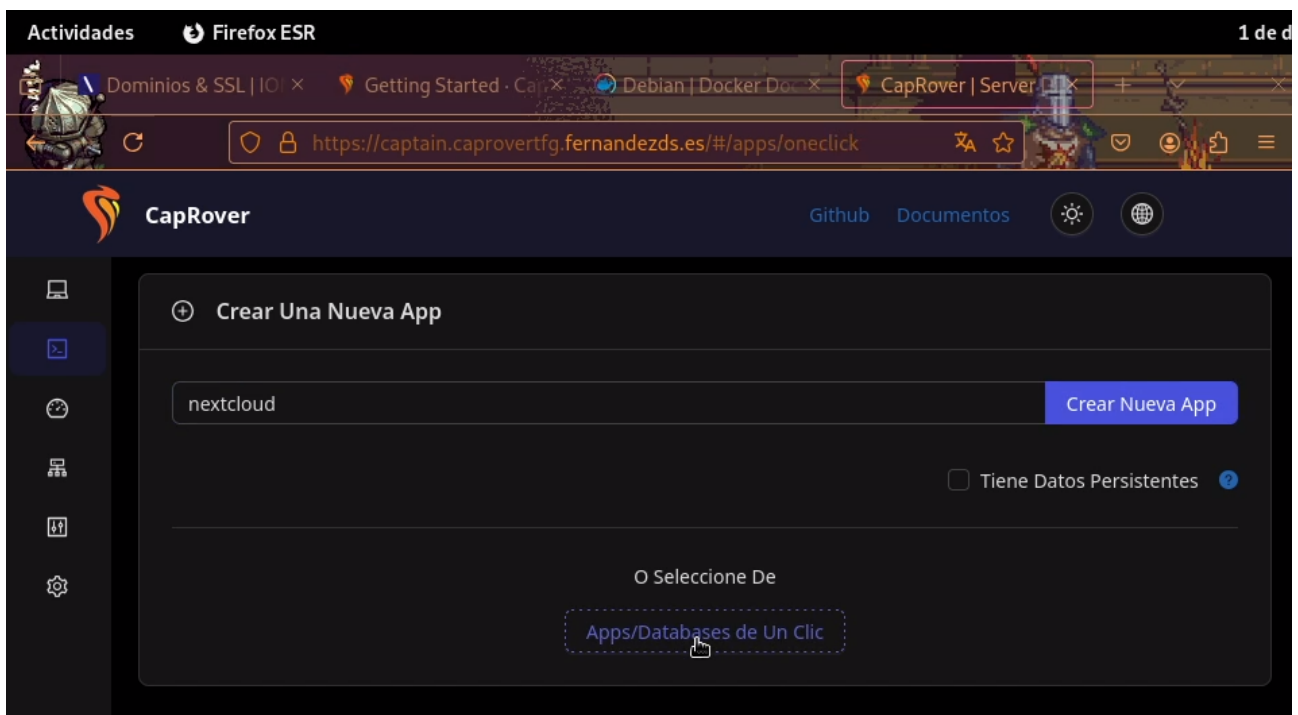
Y como se ve la aplicación **sigue funcionando**, totalmente operativa, con el nuevo ejemplo que he introducido.

### 6.1.2. Despliegue de NextCloud mediante “un clic”.

Otra opción de despliegue de aplicaciones, es mediante **el listado de aplicaciones ya predefinidas**. Esto sin duda, es el **mayor punto bajo mi punto de vista de CapRover** junto con el **despliegue en internet** de las aplicaciones.

Para este caso, **he elegido NextCloud**, una aplicación muy famosa, y **mucho más compleja que la anterior**, ya que utiliza mas de un contenedor, servicios, base de datos etc.


Para empezar, lo primero en este caso es **abrir el listado de aplicaciones en un clic**.



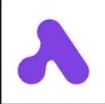
### Apps de Un Clic

Elija una app, una base de datos o un paquete (app+base de datos) de la lista a continuación. ¡El resto es magia, bueno... mago!

Las apps de un clic se obtienen del oficial [Repositorio de GitHub de Apps de Un Clic de CapRover](#) por defecto. Puede agregar otros repositorios públicos/privados si lo desea.

**Ackee**  
Self-hosted, Node.js based




**Activepieces**  
Your friendliest open source


### Apps de Un Clic

Elija una app, una base de datos o un paquete (app+base de datos) de la lista a continuación. ¡El resto es magia, bueno... mago!

Las apps de un clic se obtienen del oficial [Repositorio de GitHub de Apps de Un Clic de CapRover](#) por defecto. Puede agregar otros repositorios públicos/privados si lo desea.

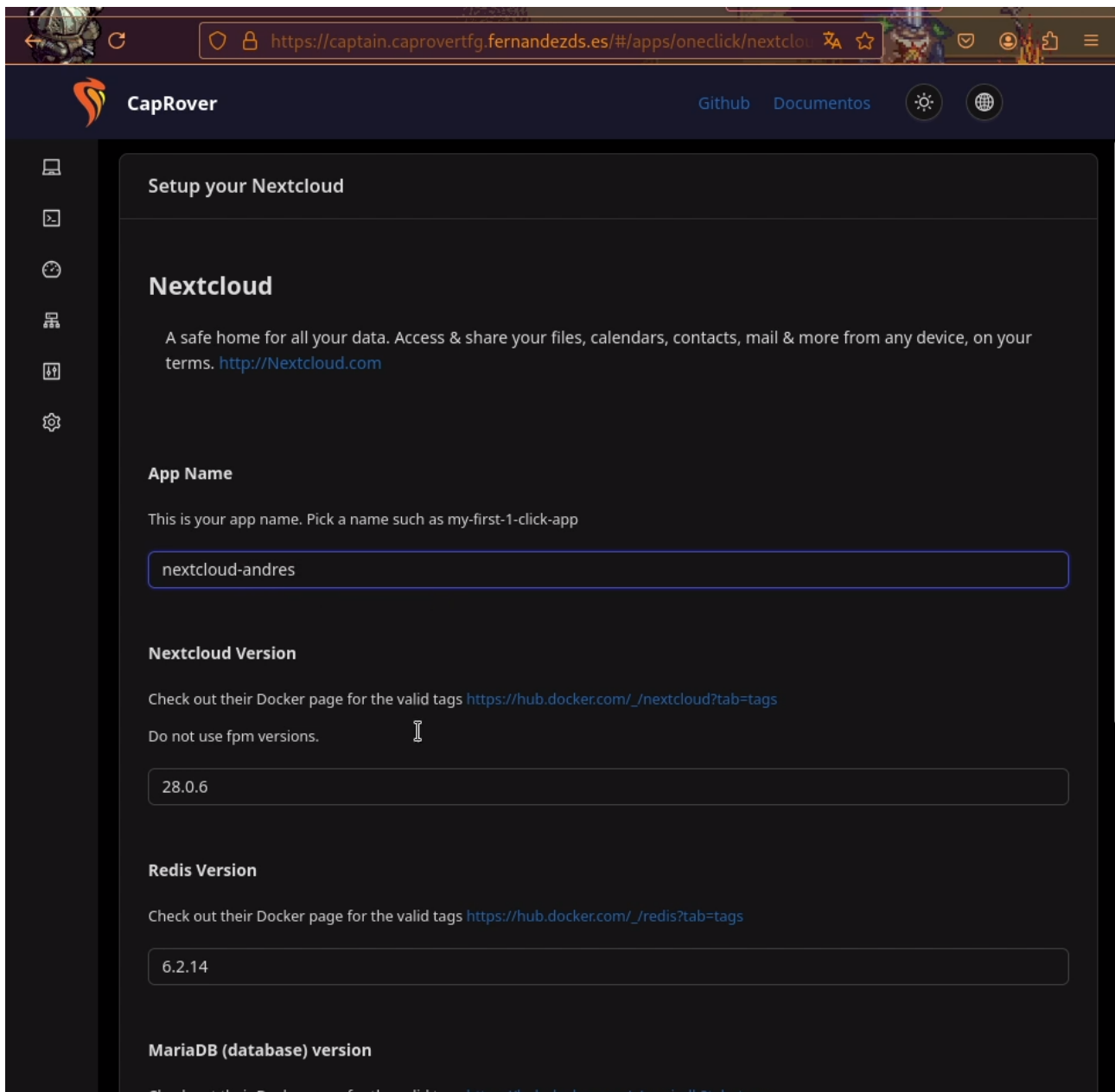
  


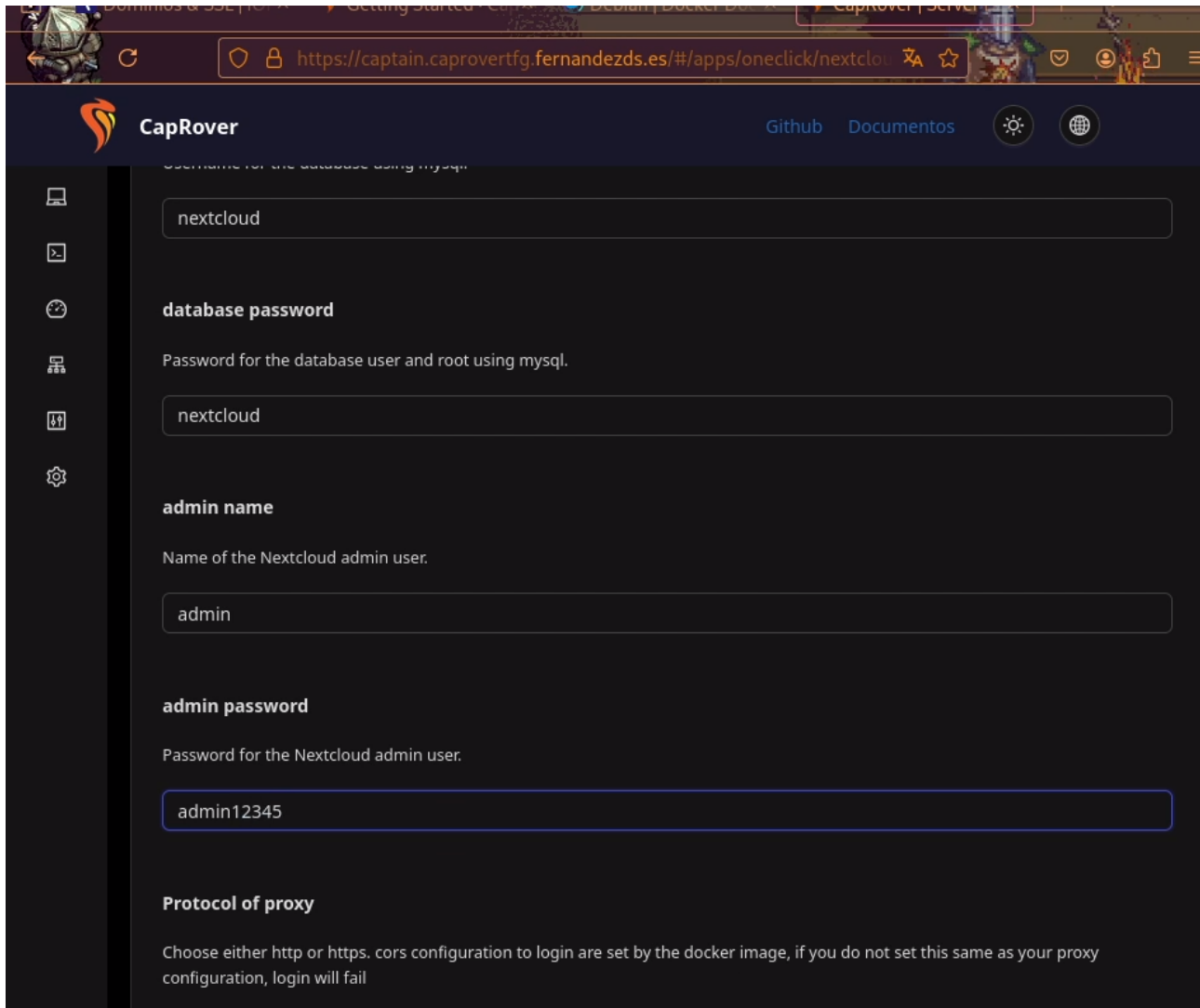
**Nextcloud**  
Nextcloud is a suite of client-server software for creating and using file hosting services



**Affine**  
Affine - There can be more than Notion and Miro. AFFINE(pronounced [ə'fain]) is a next-gen knowledge base that brings planning.

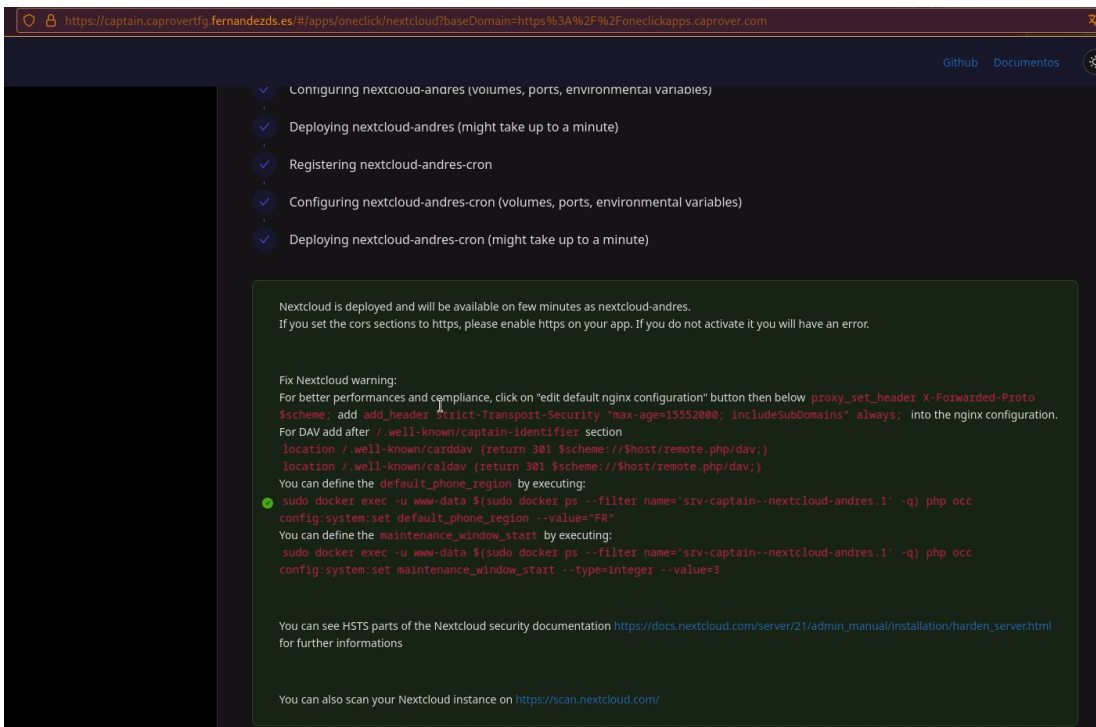
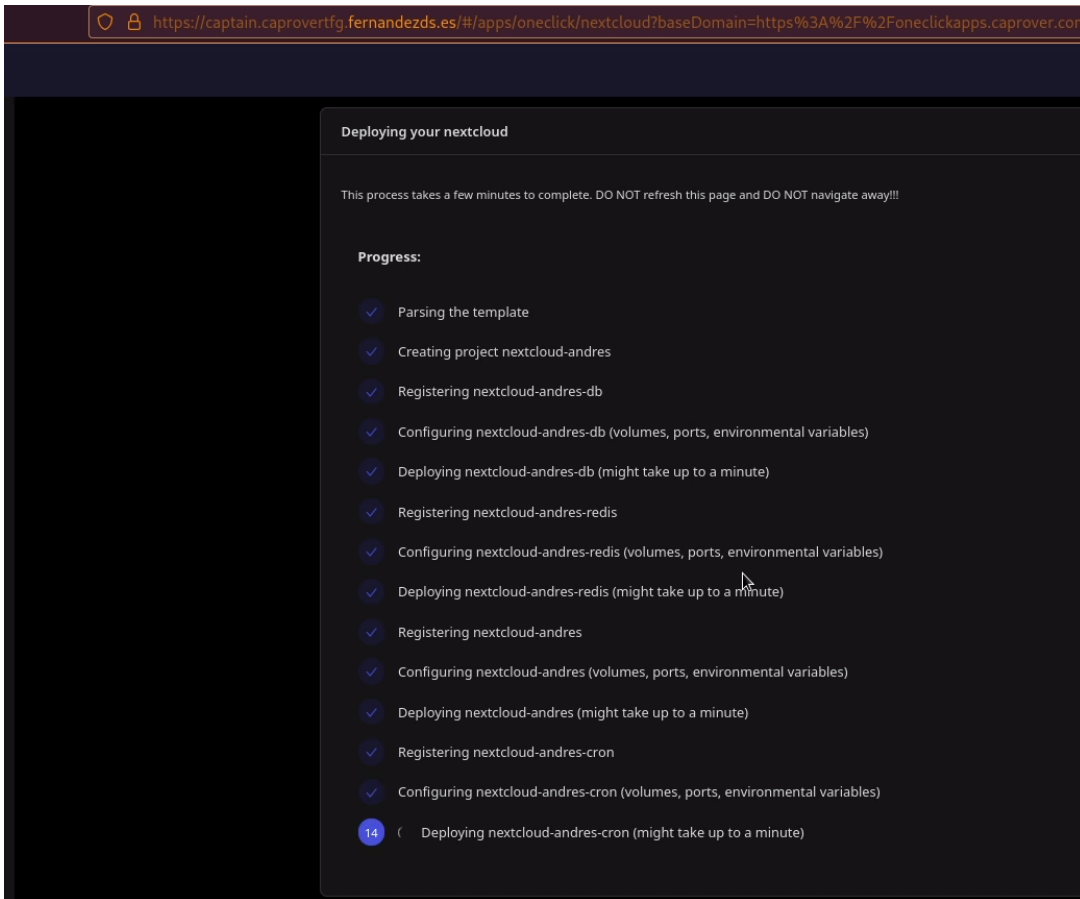
Una vez dentro, tengo que buscar en este caso, **Nextcloud**. Una vez elegida, **hay que indicarle los parámetros de la aplicación**, como un usuario, contraseña, la base de datos, etc. Se pueden elegir como por ejemplo las **versiones de las aplicaciones** de base datos que podemos elegir, entre otros parámetros que se ven en las imágenes.





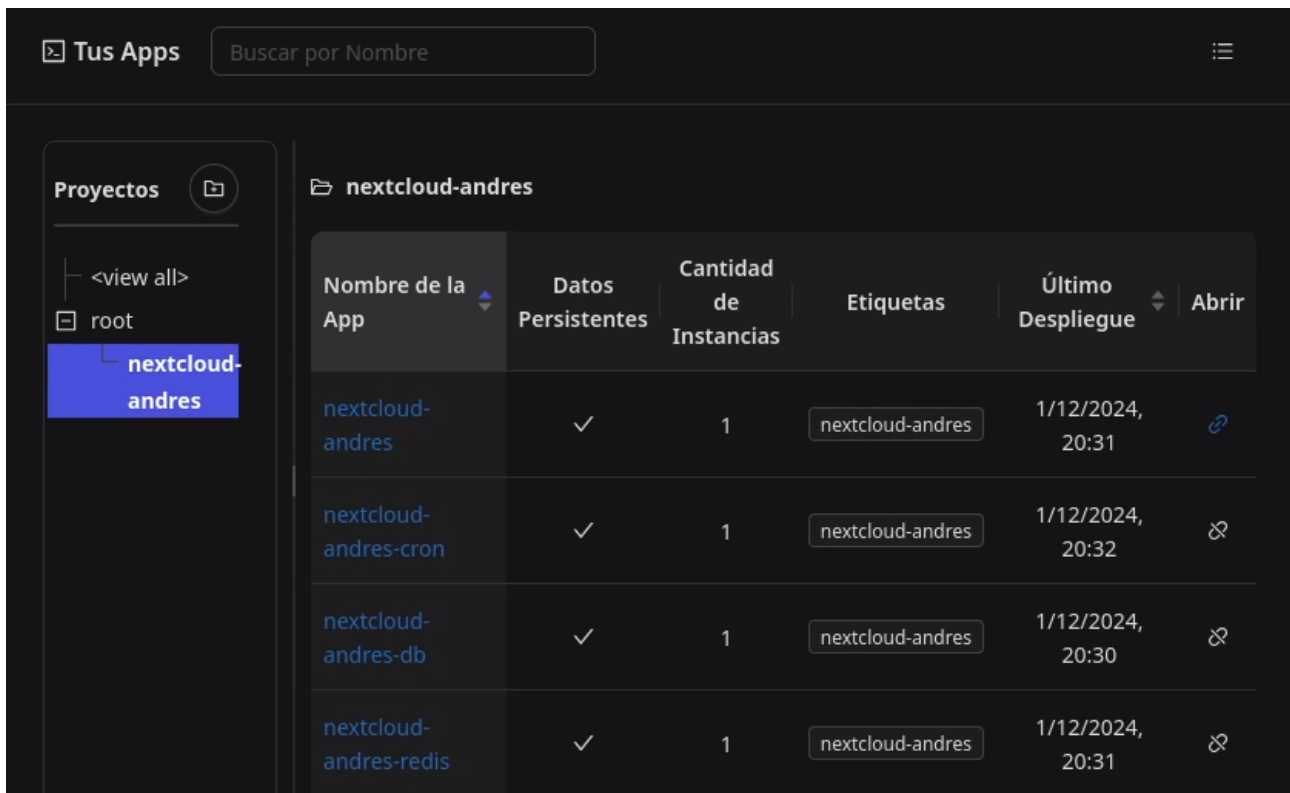
Una vez que se **hayan introducido todos los parámetros, empezará el proceso de despliegue** de la aplicación. La duración de este proceso, va adecuado a la potencia de mi máquina por lo que tarda un poco en completarse.

Este proceso de despliegue incluye **la descarga de las respectivas imágenes de Docker necesarias, la creación de los contenedores, los servicios, el despliegue de la aplicación**, etc. De esto hablaré después cuando termine.



Una vez **terminado el proceso**, nos advierte de que **podemos realizar mas configuraciones** adicionales para **mejorar u optimizar el rendimiento** de Nextcloud, de **seguridad**, y configuraciones varias, que en este caso para la demostración, no es necesario.

Como se puede comprobar en el panel de control, **Nextcloud ha creado cuatro contenedores**:



| Nombre de la App       | Datos Persistentes | Cantidad de Instancias | Etiquetas        | Último Despliegue | Abrir             |
|------------------------|--------------------|------------------------|------------------|-------------------|-------------------|
| nextcloud-andres       | ✓                  | 1                      | nextcloud-andres | 1/12/2024, 20:31  | <a href="#">🔗</a> |
| nextcloud-andres-cron  | ✓                  | 1                      | nextcloud-andres | 1/12/2024, 20:32  | <a href="#">🔗</a> |
| nextcloud-andres-db    | ✓                  | 1                      | nextcloud-andres | 1/12/2024, 20:30  | <a href="#">🔗</a> |
| nextcloud-andres-redis | ✓                  | 1                      | nextcloud-andres | 1/12/2024, 20:31  | <a href="#">🔗</a> |

- **Nextcloud-andres:** Donde se aloja la aplicación principal como tal.
- **Nextcloud-andres-cron:** Donde tienen lugar las tareas de cron como notificaciones, índices, sincronización etc.
- **Nextcloud-andres-db:** Donde se encuentra la base de datos de la aplicación, pero para datos persistentes como por ejemplo usuarios, configuraciones etc.

- **Nextcloud-andres-redis:** También como base de datos pero para la memoria caché y las sesiones para ayudar y reducir la carga de la base de datos principal digamos (suponiendo que haya mucho contenido u usuarios).

Esto en la máquina VPS servidor, es lo mismo. **Se han creado los contenedores y los servicios:**

```
docker service ls
```

| NAME  | MODE       | REPLICAS | IMAGE        |
|---|------------|----------|--------------|
| captain-captain<br>caprover : 1.13.2                          | replicated | 1/1      | caprover/    |
| captain-certbot<br>certbot-sleeping:v2.11.0                   | replicated | 1/1      | caprover/    |
| captain-nginx   | replicated | 1/1      | nginx:1.24   |
| srv-captain--nextcloud-andres<br>nextcloud:28.0.6             | replicated | 1/1      |              |
| srv-captain--nextcloud-andres-cron<br>nextcloud-andres-cron:1 | replicated | 1/1      | img-captain- |
| srv-captain--nextcloud-andres-db<br>mariadb:10.6.18           | replicated | 1/1      |              |
| srv-captain--nextcloud-andres-redis                           | replicated | 1/1      | redis:6.2.14 |

```
docker ps
```

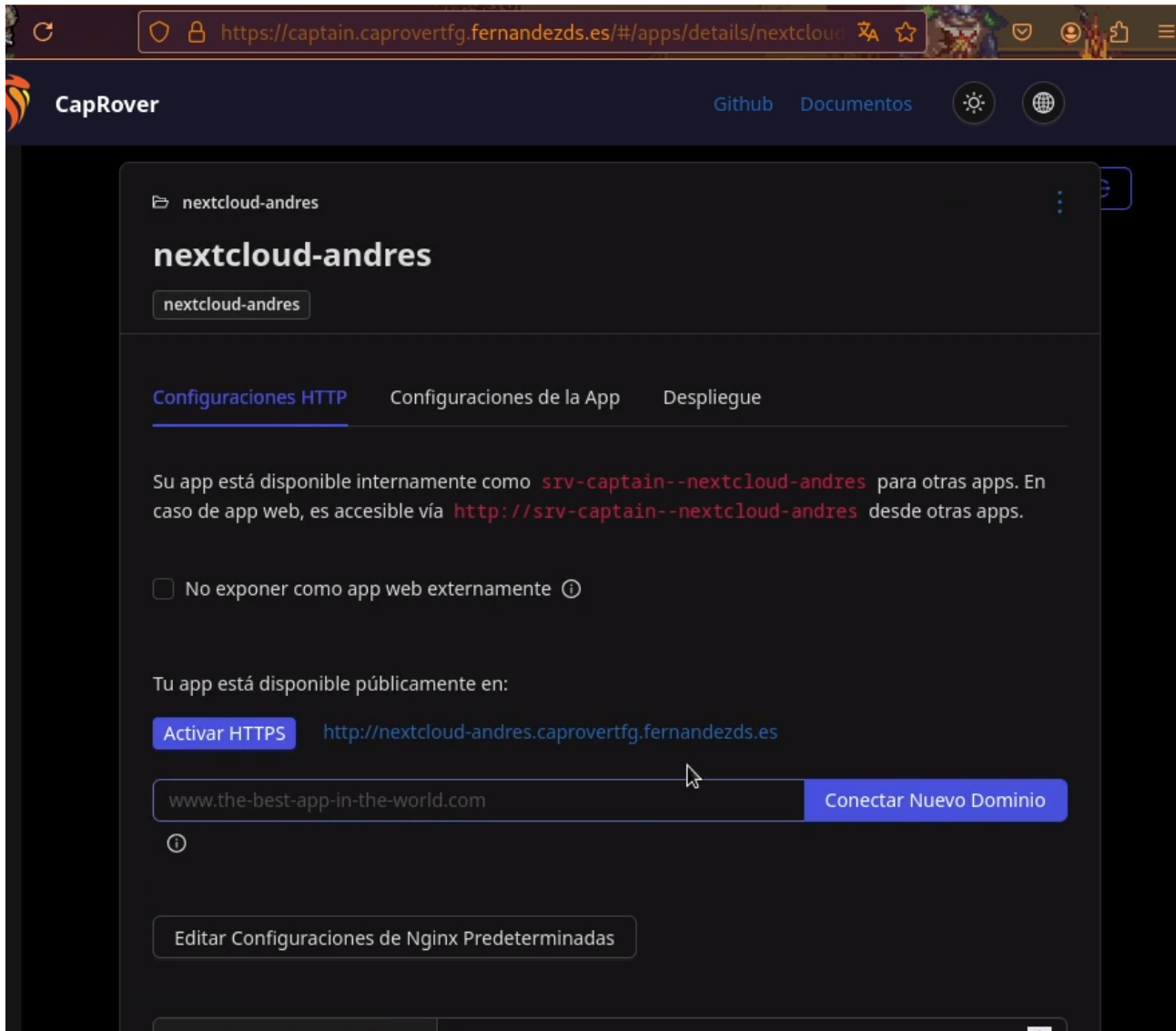
| IMAGE                               | STATUS        | NAMES                               | COMMAND                  | CREATED        |
|-------------------------------------|---------------|-------------------------------------|--------------------------|----------------|
| nextcloud:28.0.6                    | Up 17 minutes | srv-captain--nextcloud-andres       | "/entrypoint.sh apac..." | 17 minutes ago |
| img-captain-nextcloud-andres-cron:1 | Up 23 minutes | srv-captain--nextcloud-andres-cron  | "/cron.sh"               | 23 minutes ago |
| redis:6.2.14                        | Up 24 minutes | srv-captain--nextcloud-andres-redis | "docker-entrypoint.s..." | 24 minutes ago |
| mariadb:10.6.18                     | Up 39 minutes | srv-captain--nextcloud-andres-db    | "docker-entrypoint.s..." | 39 minutes ago |
| caprover/certbot-sleeping:v2.11.0   | Up 39 minutes | captain-certbot.1.gui4qrb           | "/bin/sh -c 'sleep 9..." | 39 minutes ago |
| nginx:1.24                          | Up 40 minutes | captain-nginx.1.u4zdb3h             | "docker-entrypoint...."  | 40 minutes ago |

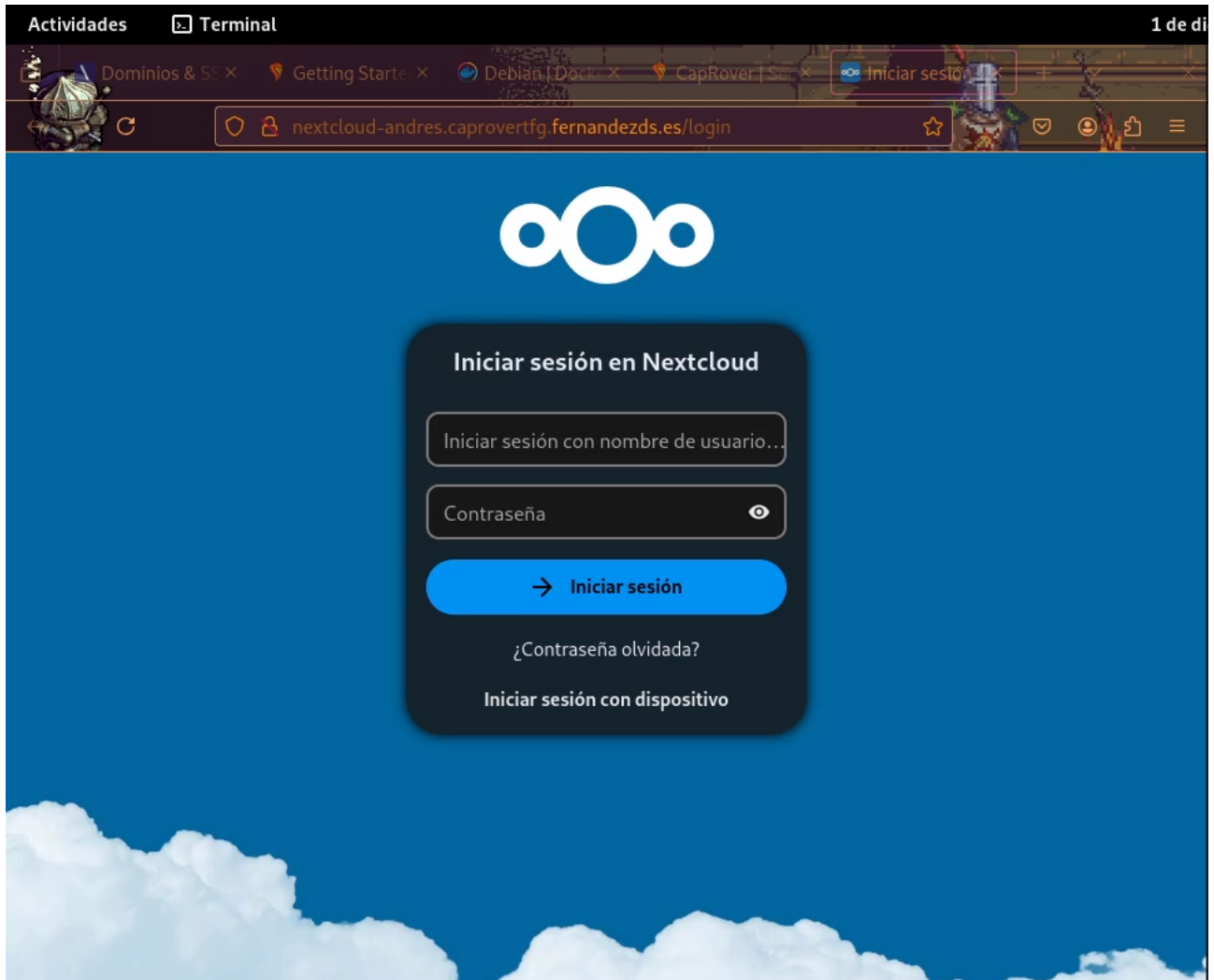


Andrés Fernández de Santaella

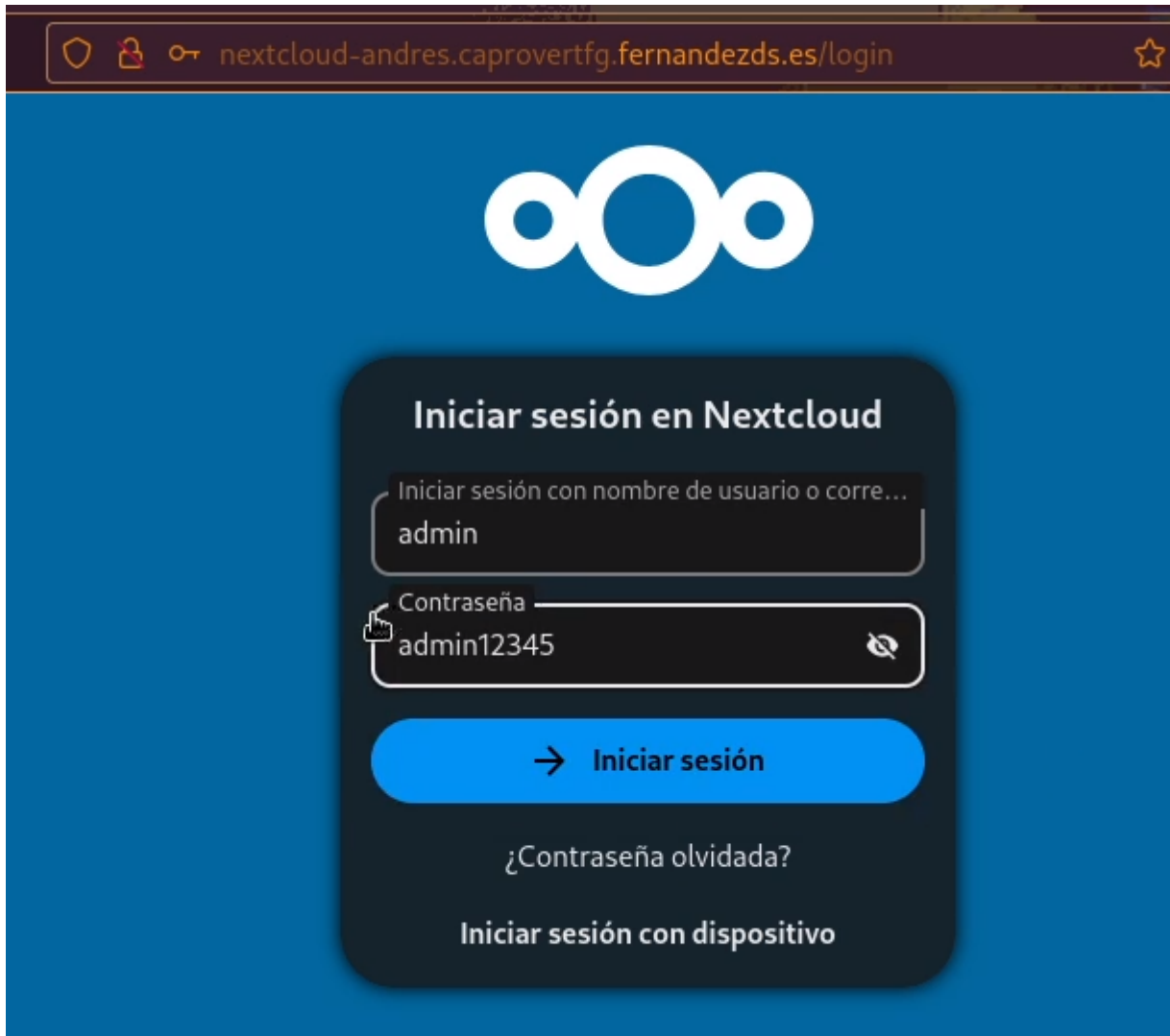
caprover/caprover:1.13.2 "docker-entrypoint.s..." 40 minutes ago  
Up 40 minutes captain-captain.1.0guaj3t

Al entrar en la **aplicación principal**, es donde nos indica **donde se ha desplegado la aplicación**.

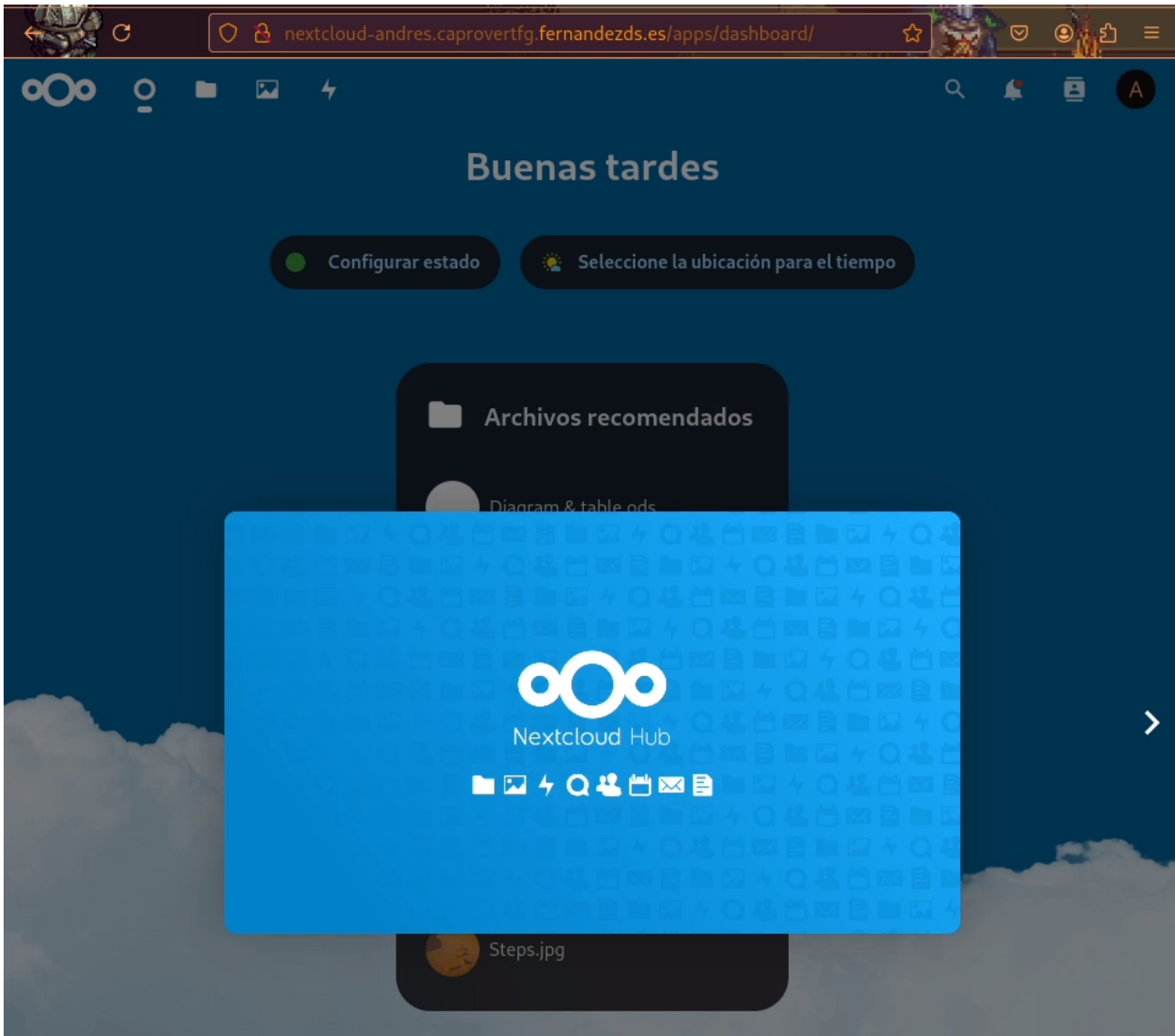




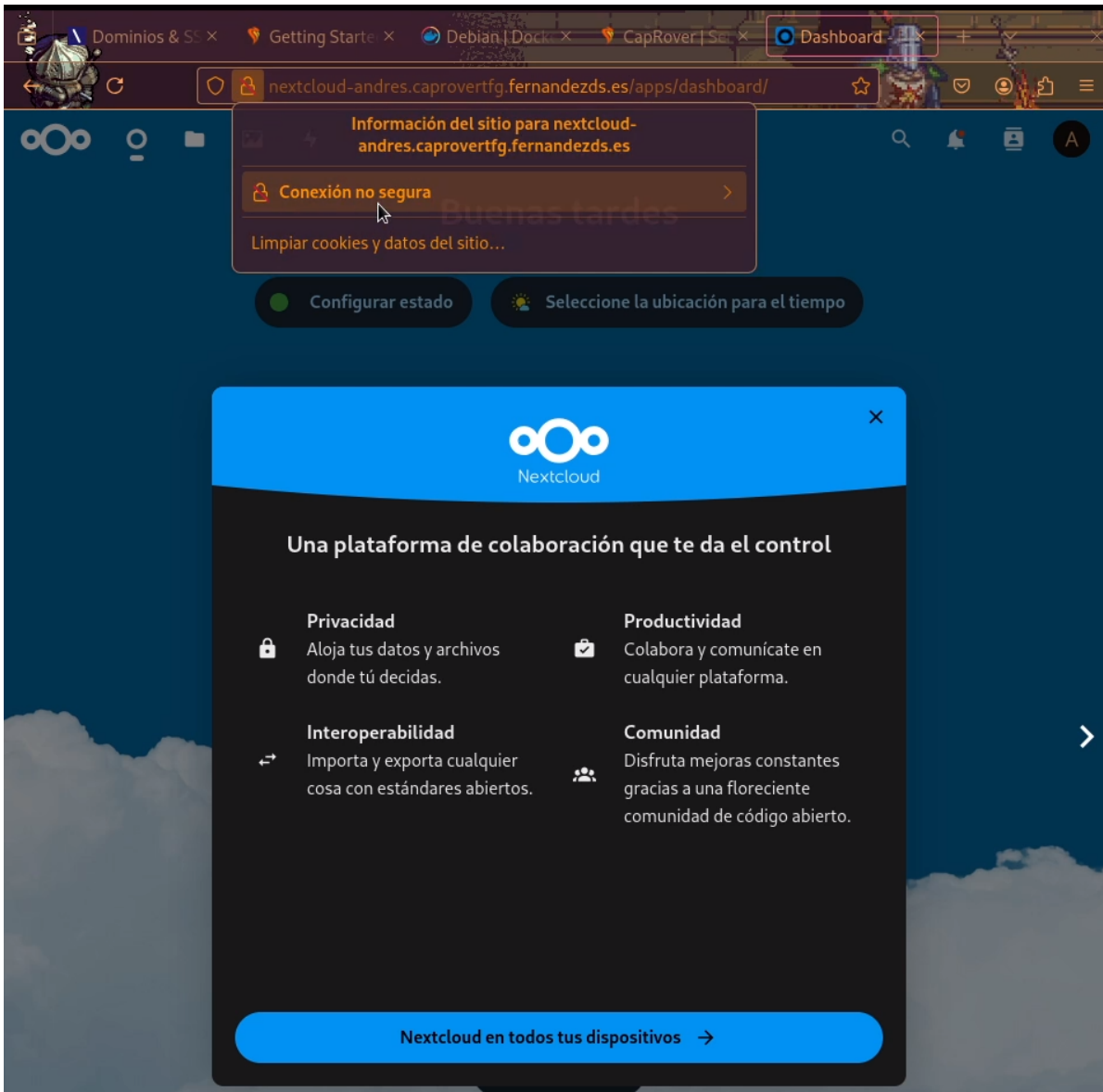
Como se ve en las imágenes, el **despliegue se ha realizado con éxito**, y la aplicación ya se encuentra lista para funcionar.



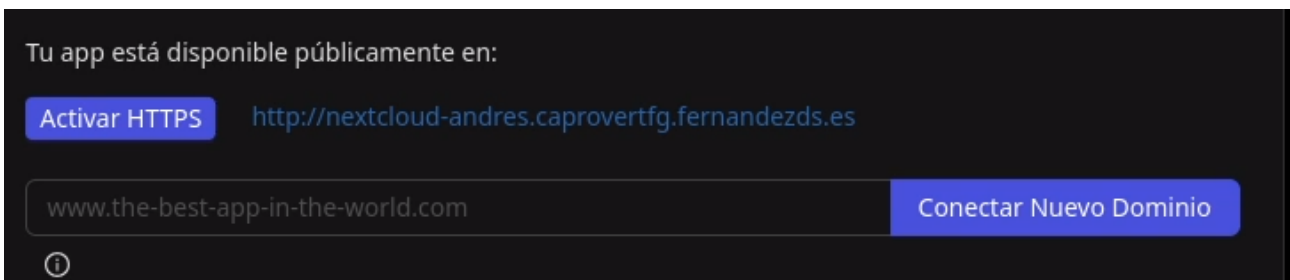
Para acceder es con el **usuario y la contraseña que especifiqué al principio**, cuando había que especificar los parámetros de la aplicación.



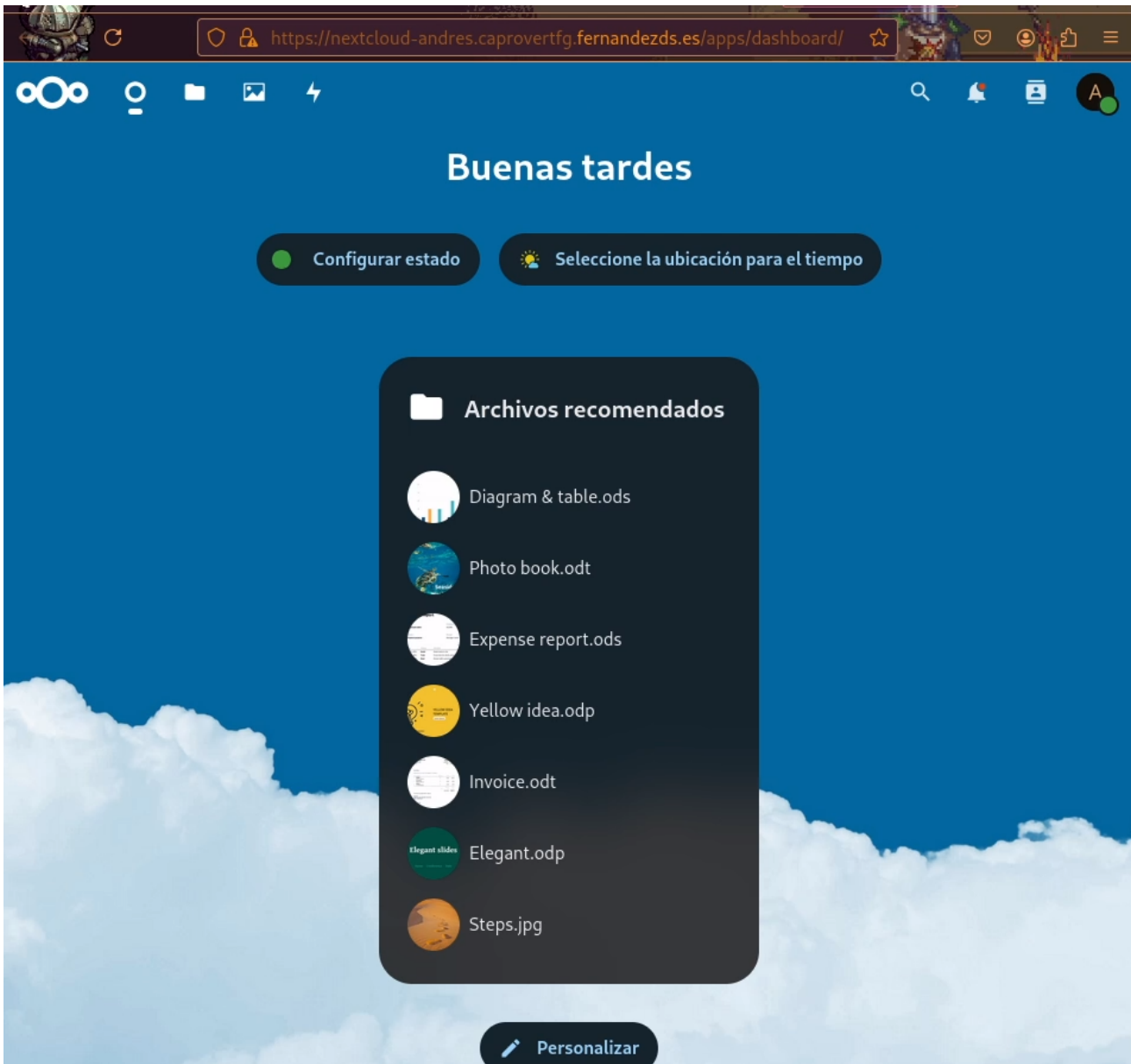
Aunque la aplicación ya está completamente funcional para su uso, **falta aplicarle el modo HTTPS**, para dejarla completamente finalizada para su uso.

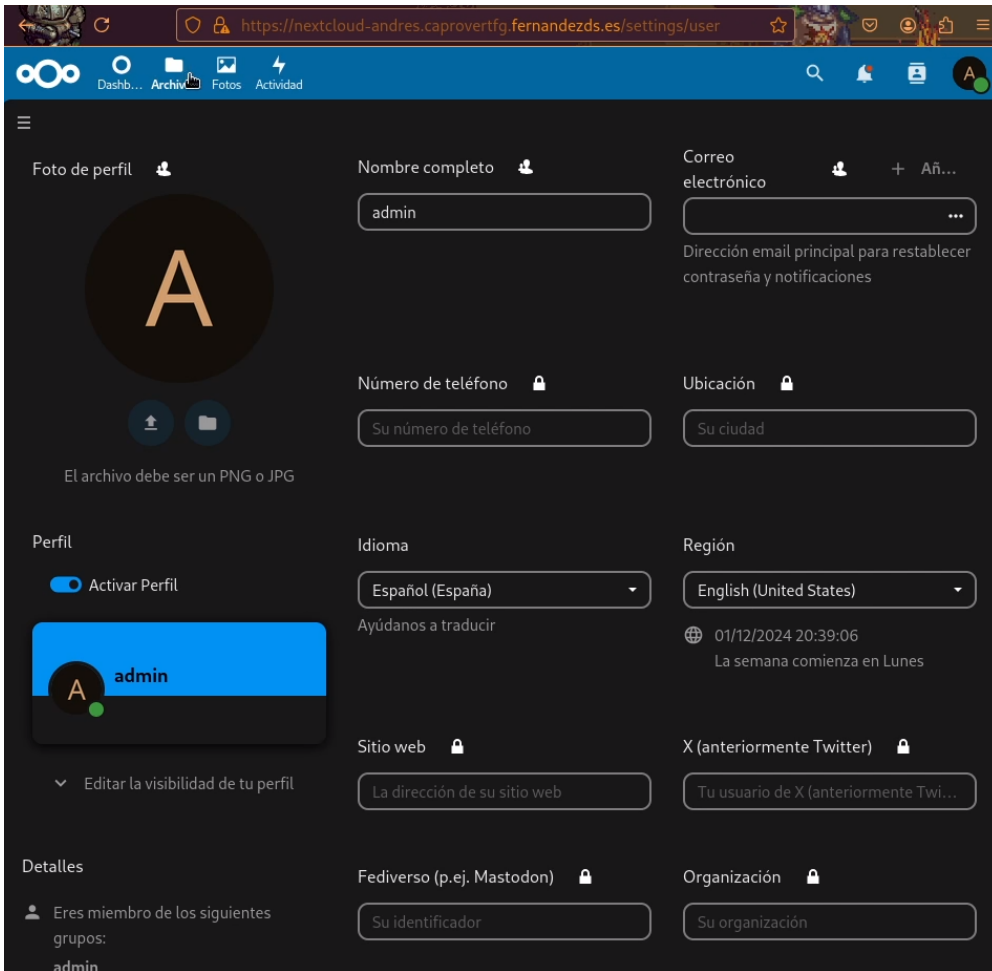


Y para ello, es con el **mismo procedimiento que con el resto de aplicaciones**, activando el botón de HTTPS:



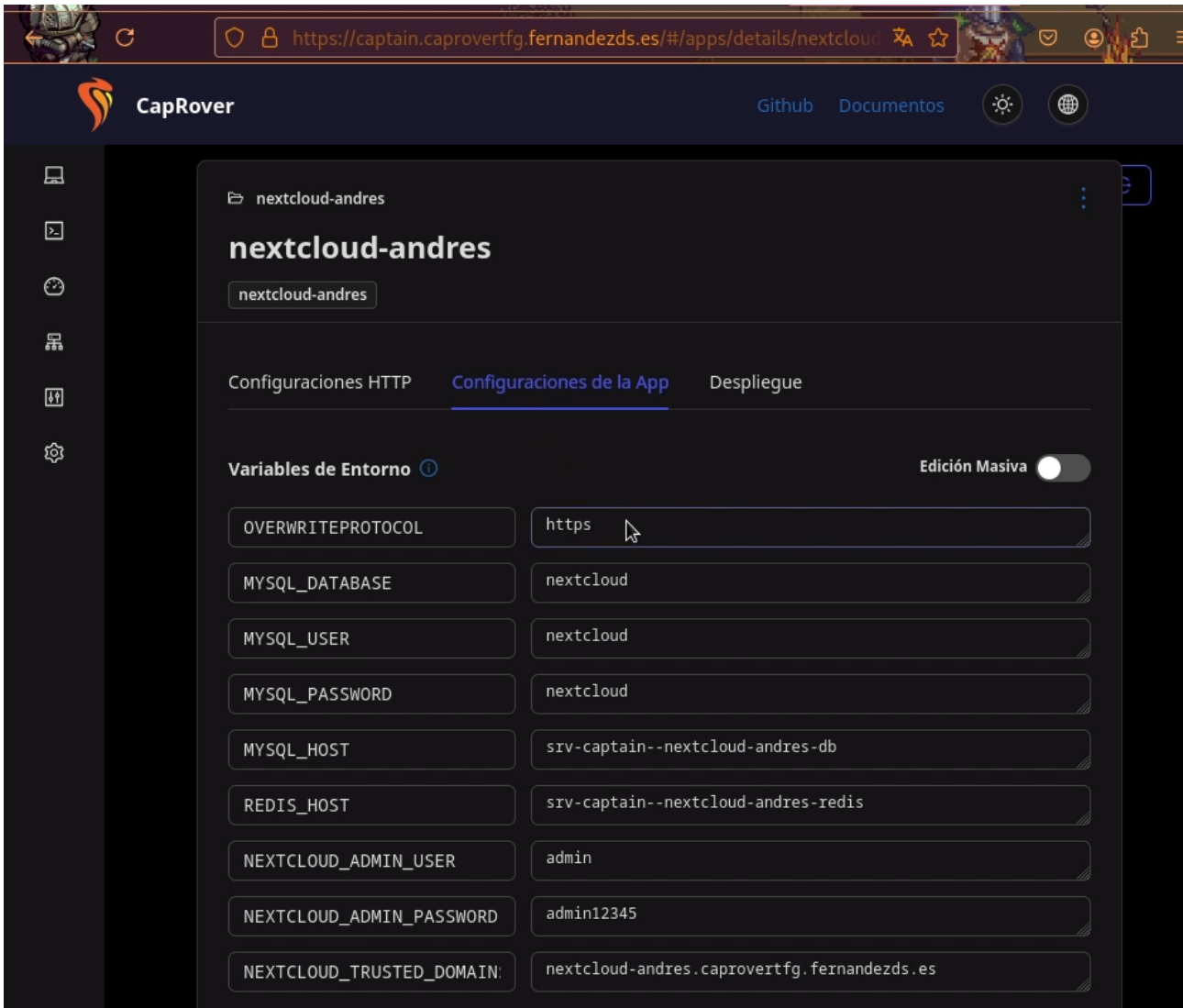
Hecho esto la aplicación **ya funciona perfectamente con HTTPS.**





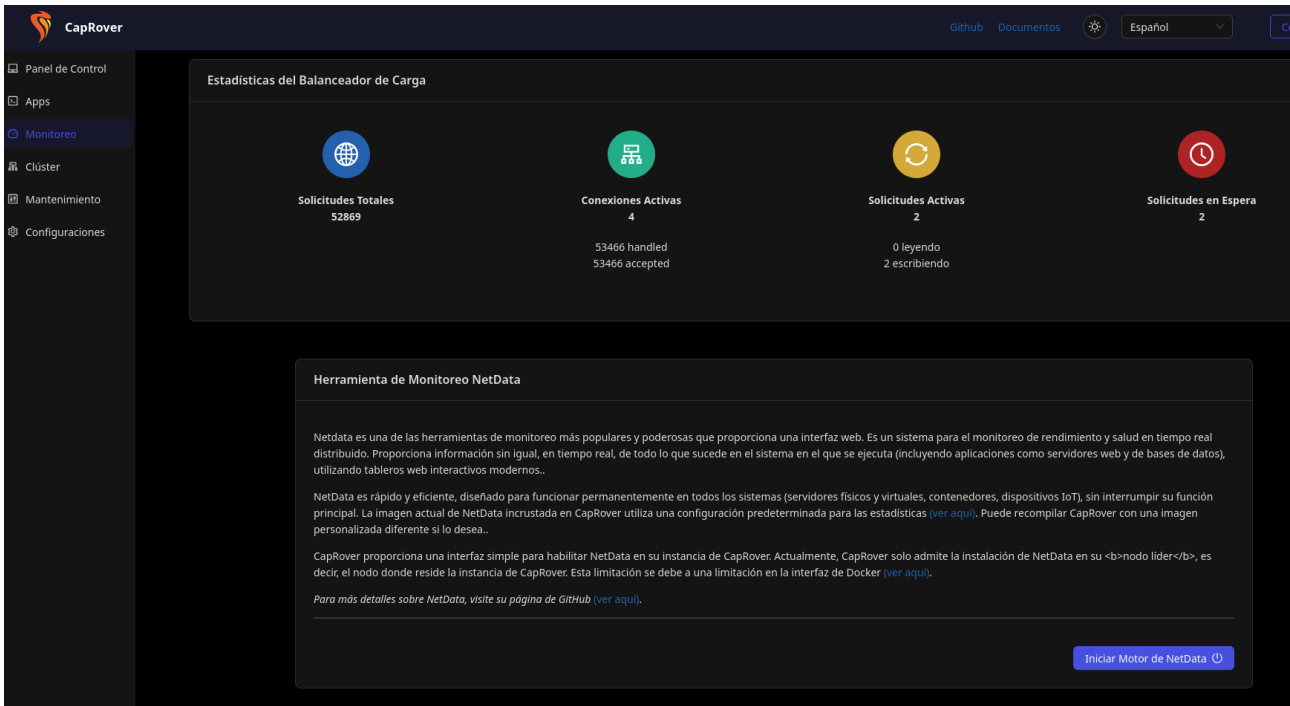
Aparece un **pequeño aviso con el modo https**, pero eso ha sido **un error por mi parte** ya que al inicio de la configuración de la aplicación **la marqué como https, en vez de http**. Por lo que el aviso sigue apareciendo pero es por este motivo.

Al desplegarse las aplicaciones, **estas lo hacen en modo HTTP**, y luego se le activa el modo **HTTPS manualmente**.

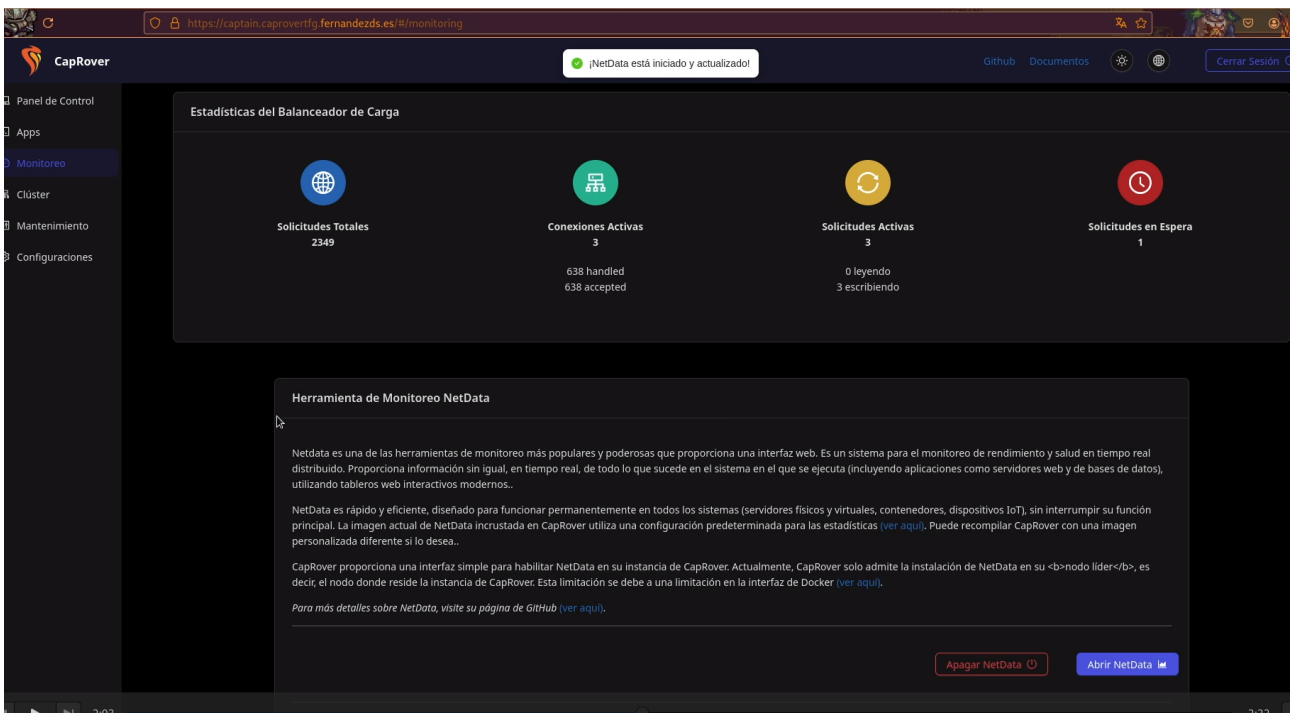


Otro punto interesante para comprobar es **el monitoreo del sistema**, CapRover ofrece esta posibilidad **mediante NetData**, directamente desde la **interfaz Web**. Esto es super útil, ya que podemos ver en todo momento como esta la máquina, en el caso de que **no tengamos acceso en ese momento al servidor**.



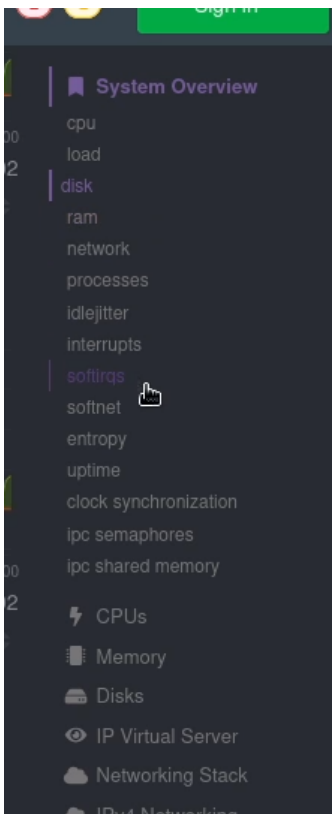
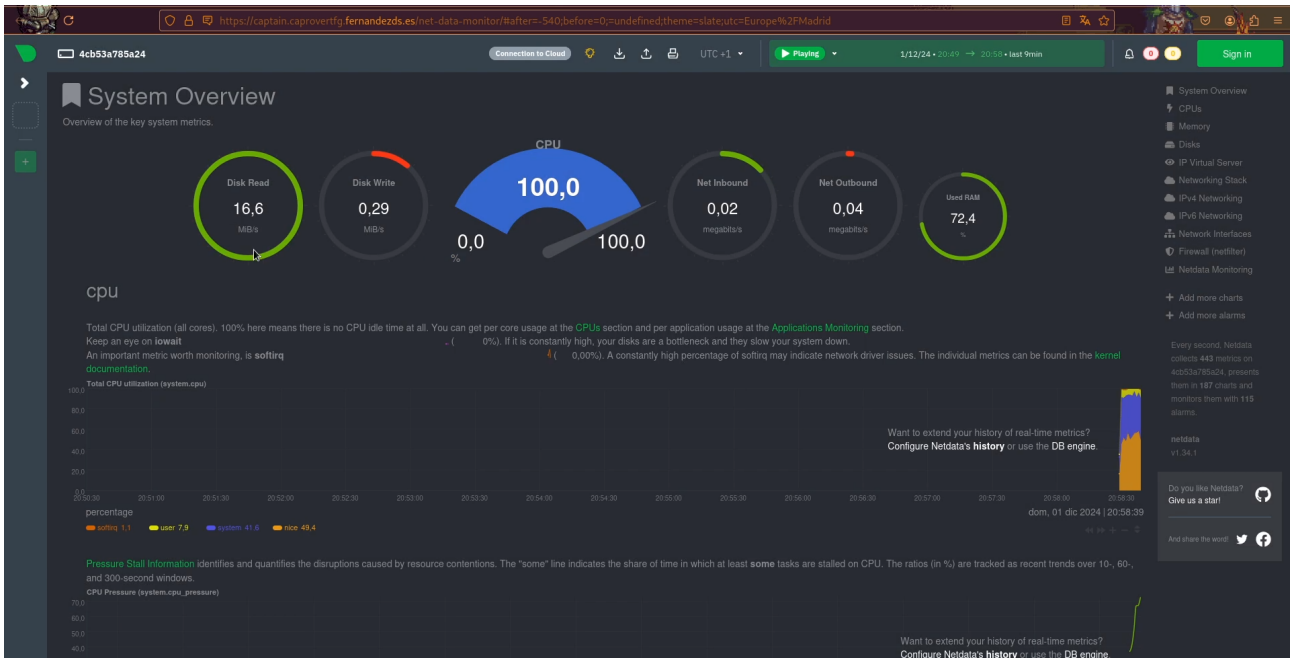


Este servicio en realidad, **se trata del despliegue de otro contenedor que contiene NetData**, y para poder usarlo hay que desplugarlo manualmente en “Iniciar monitor de NetData”.



Andrés Fernández de Santaella

Una vez desplegado, se abre una **nueva pestaña con la web de la aplicación ya desplegada** donde podemos comprobar muchos, pero muchos valores del estado de la máquina, uso de CPU, uso de memoria, lectura, escritura, y **todo lo que ofrece el monitor de NetData**.



Si **compruebo los contenedores del servidor VPS**, puedo verificar como **se ha desplegado un nuevo contenedor** para el uso de NetData como ya he mencionado.

| IMAGE                    | COMMAND           | CREATED       | STATUS            |
|--------------------------|-------------------|---------------|-------------------|
| caprover/netdata:v1.34.1 | "/run.sh"         | 2 minutes ago | Up About a minute |
| PORTS                    | NAMES             |               |                   |
| 19999/tcp                | netdata-container |               |                   |

Y de esta forma, **es posible instalar aplicaciones de un clic como Nextcloud** en esta demostración. Pese a la cantidad de imágenes, el proceso es muy sencillo. **Se elige aplicación, se introducen los parámetros necesarios** (varían dependiendo de la aplicación), **y esperar a que se despliegue.**

Esto **suenas muy bonito y muy fácil en la teoría**, pero **en la practica para que realmente sea así, la máquina servidor debe estar en buen funcionamiento y bien configurada**, y sobretodo que tenga los recursos necesarios para desplegar la aplicación.

De no contar con ello, pasará lo que me pasó durante las pruebas, **la aparición de errores muy extraños en la máquina, que no solo se limitan a saturarla.** De esto hablaré con más profundidad en la sección de errores.

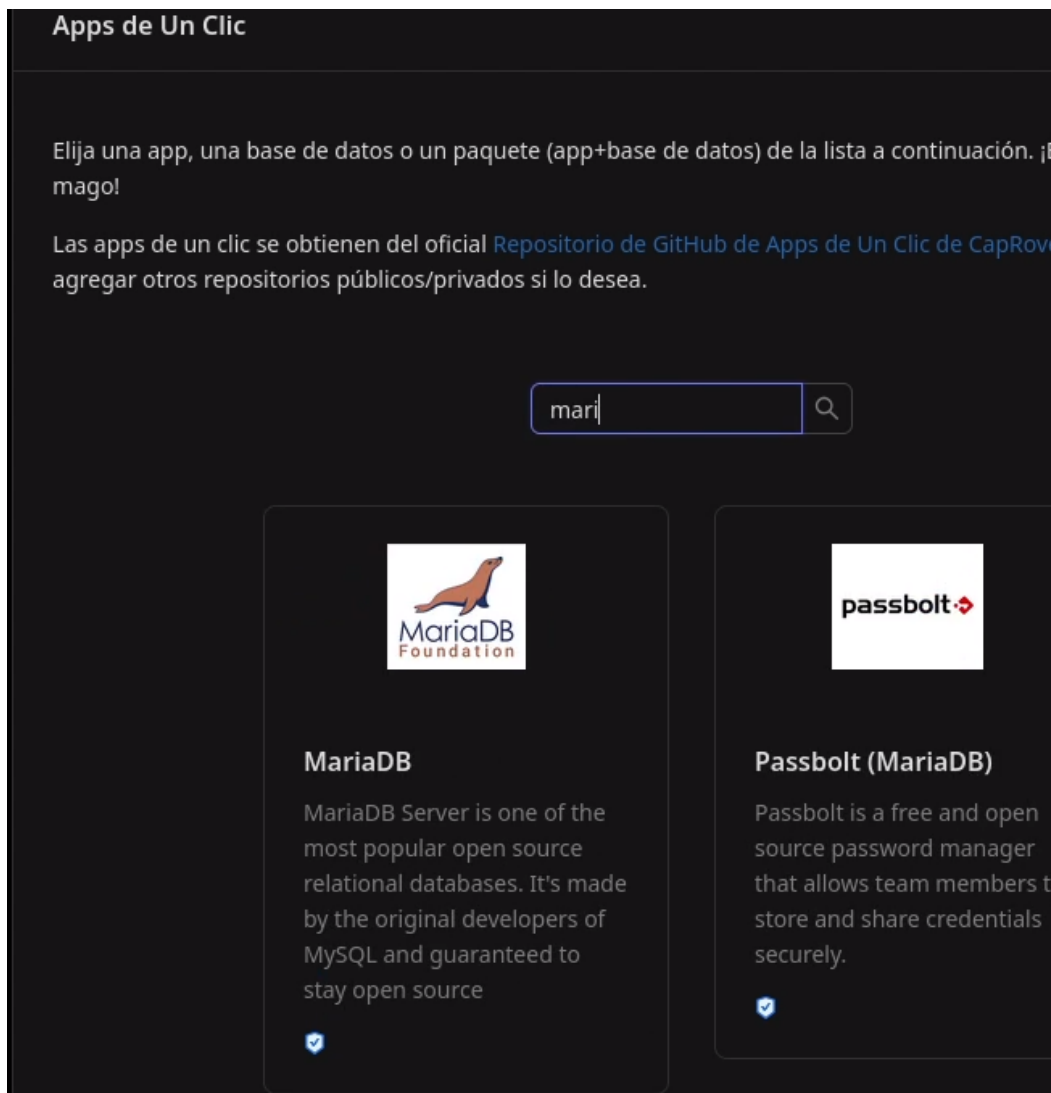
## 6.2. Despliegue Interconectando Aplicaciones.

Para esta demostración, voy a explicar como **desplegar una aplicación que necesita de una segunda para funcionar**, y que no este disponible en las aplicaciones de un clic. Por ejemplo, voy a desplegar **Wordpress y MariaDB por separado.** (La versiones de Wordpress disponibles son dos, una que contiene todo lo necesario digamos "All In One" y otra que sólo es la propia aplicación, como si la descargásemos de su página oficial).

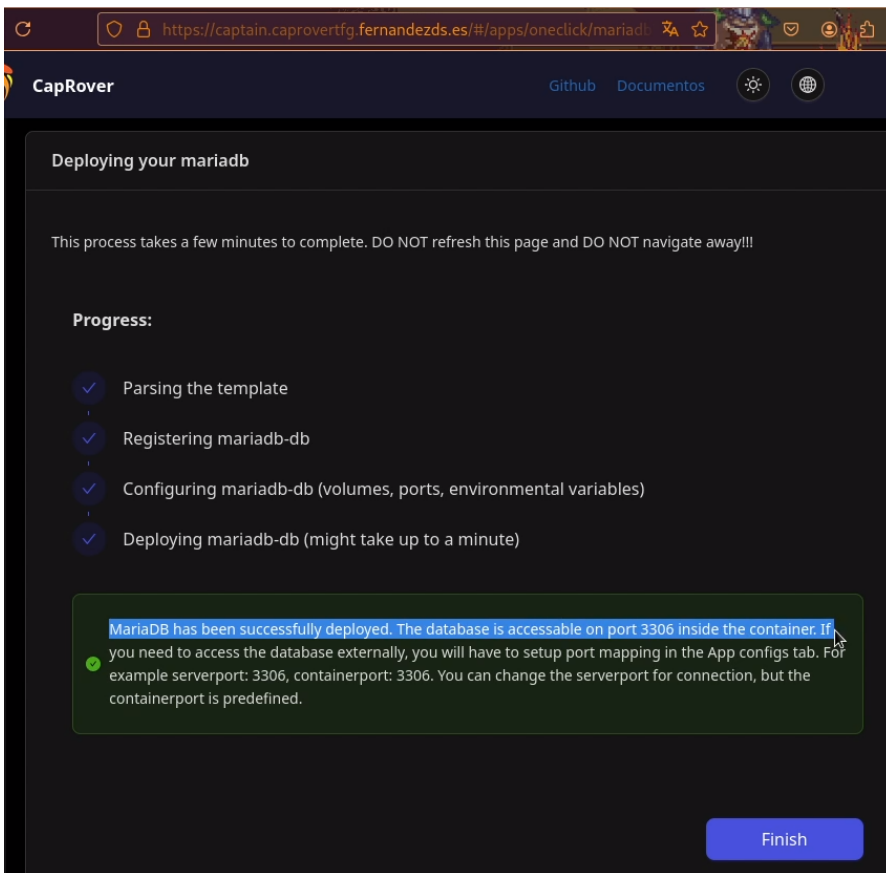
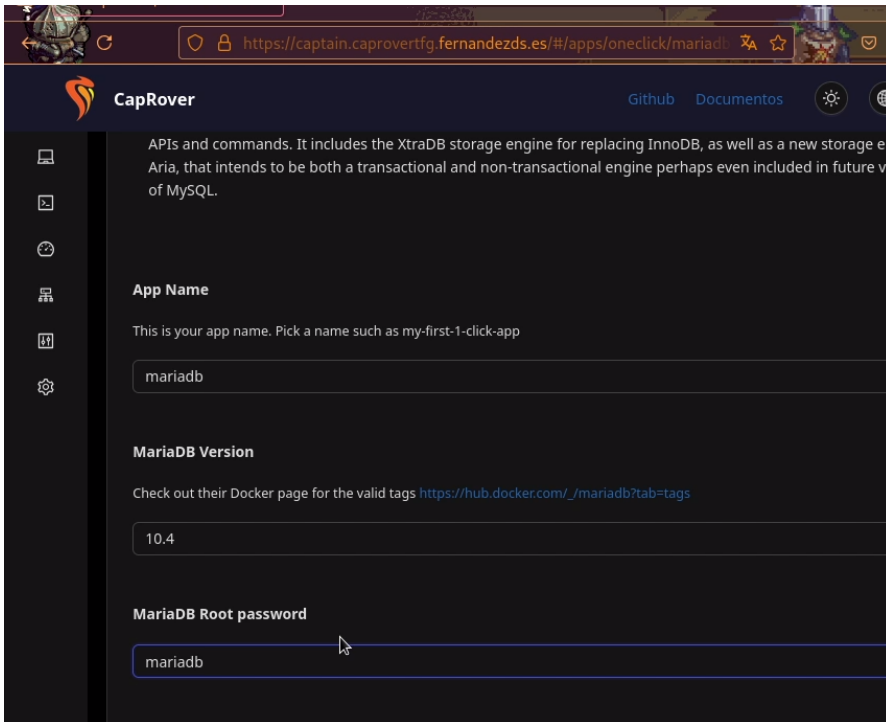
Antes de empezar, hay que aclarar que **cada vez que se despliega una aplicación, CapRover le asigna un nombre identificativo** a nivel interno, precisamente para identificar y por si tenemos que interconectar dichas aplicaciones.

### 6.2.1 Despliegue de Wordpress.

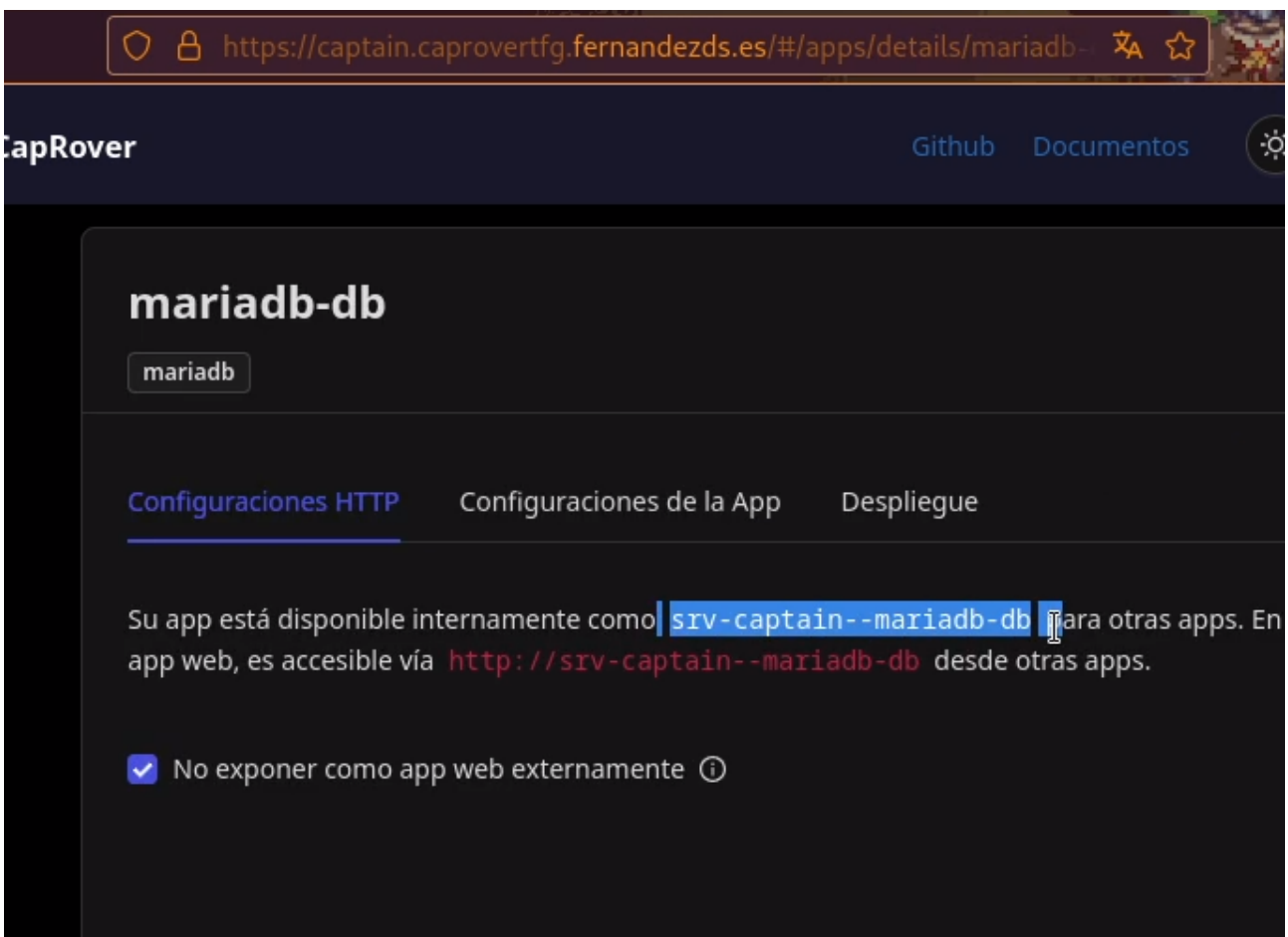
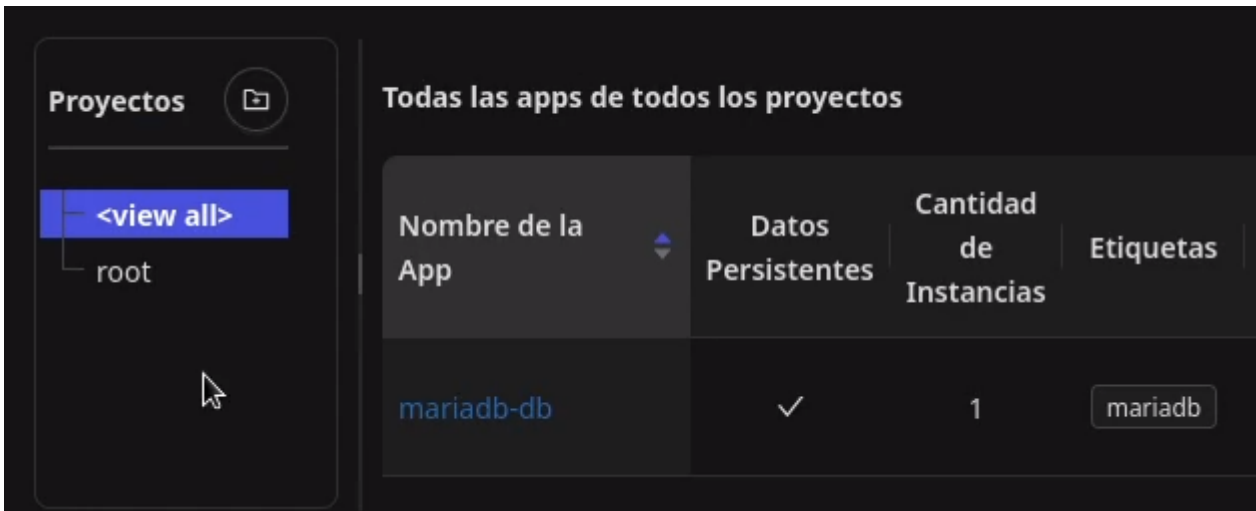
El primer paso, es **desplegar la base de datos, en este caso Mariadb**, y para ello hago uso de la lista de aplicaciones disponibles.



Le **indico las variables** como el nombre de la aplicación, la contraseña de root, etc.

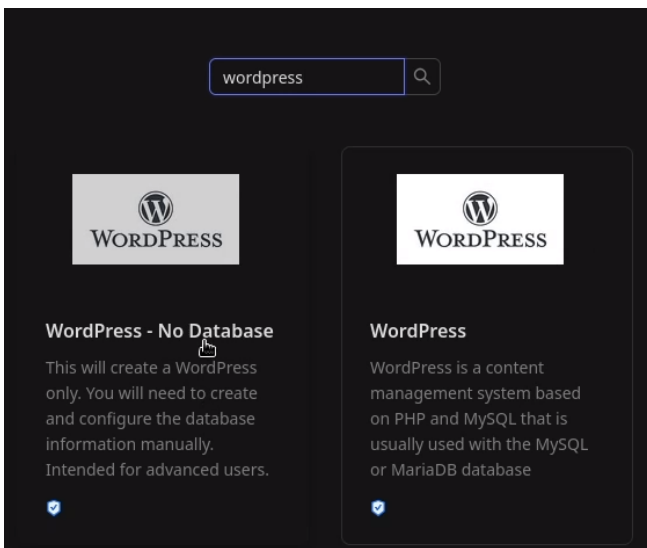


Una vez desplegada, ya aparece en el listado de proyectos de CapRover, para poder gestionarla a placer. Ahora lo que me interesa saber es que nombre se le ha asignado.



Como he mencionado antes, **con cada despliegue se le asigna un nombre identificativo** a cada aplicación, en este caso para la aplicación MariaDB, es **srv-captain—mariadb-db**. Esto es importante **para poder especificarle a la aplicación Wordpress, donde se aloja la base de datos** con la que va a trabajar.

Con la base de datos ya preparada, es momento de **desplegar la versión de Wordpress que no incluye base de datos** para poder demostrar el funcionamiento. Para ello, sólo la busco en el listado de aplicaciones, y elijo la que especifica **“No Database”**.



**App Name**  
This is your app name. Pick a name such as my-first-1-click-app

wordpress

**Database Host**  
Database host

srv-captain—mariadb-db

**DB Name**  
Database name

wordpress

**Table Prefix**  
Table prefix used by this Wordpress

WP\_ADMIN

**Database user**

mariadb

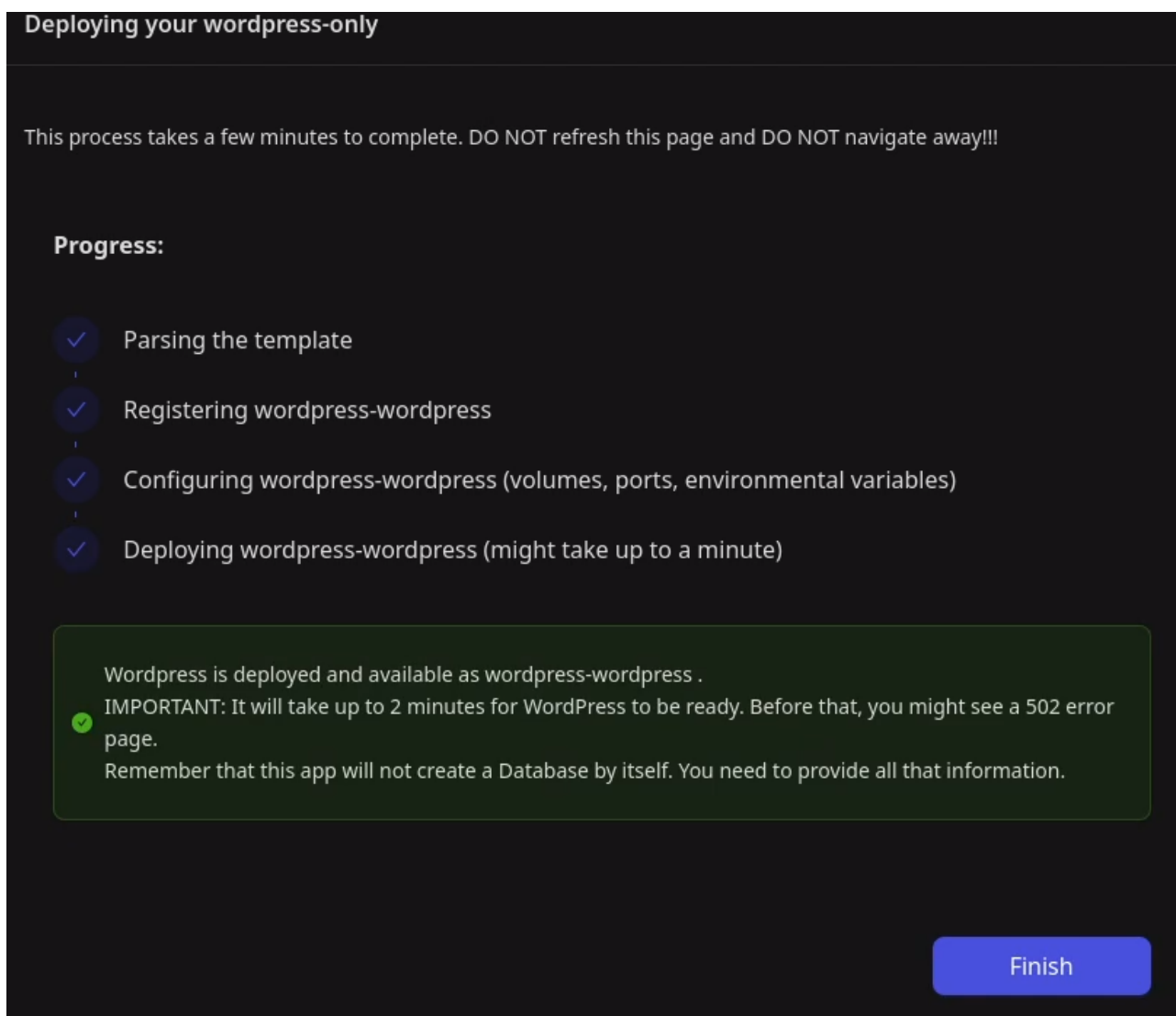
**Database password**

mariadb

Andrés Fernández de Santaella

Como siempre, **se rellenan los campos con los datos necesarios** como nombre de la base de datos, el usuario, el nombre de la aplicación, etc.

Y una vez introducidas, **empieza el proceso de desligue de la propia aplicación**. Una vez terminado, nos indica que esperemos un par de minutos para que este totalmente desplegada.



The screenshot shows a dark-themed interface for a deployment process. At the top, it says "Deploying your wordpress-only". Below that, a warning message states: "This process takes a few minutes to complete. DO NOT refresh this page and DO NOT navigate away!!!". A "Progress:" section lists four steps, each with a blue checkmark in a circle: "Parsing the template", "Registering wordpress-wordpress", "Configuring wordpress-wordpress (volumes, ports, environmental variables)", and "Deploying wordpress-wordpress (might take up to a minute)". Below the list is a green-bordered box containing the text: "Wordpress is deployed and available as wordpress-wordpress . IMPORTANT: It will take up to 2 minutes for WordPress to be ready. Before that, you might see a 502 error page. Remember that this app will not create a Database by itself. You need to provide all that information." At the bottom right, there is a blue button labeled "Finish".

En este caso, **aunque esperemos la aplicación no va a funcionar**, porque **no se le ha especificado donde se encuentra la base de datos**. Para ello debo entrar a la aplicación he introducirle de nuevo las variables para que cargue correctamente.



Todas las apps de todos los proyectos

| Nombre de la App                 | Datos Persistentes | Cantidad de Instancias | Etiquetas              |
|----------------------------------|--------------------|------------------------|------------------------|
| <code>mariadb-db</code>          | ✓                  | 1                      | <code>mariadb</code>   |
| <code>wordpress-wordpress</code> | ✓                  | 1                      | <code>wordpress</code> |

### mariadb-db

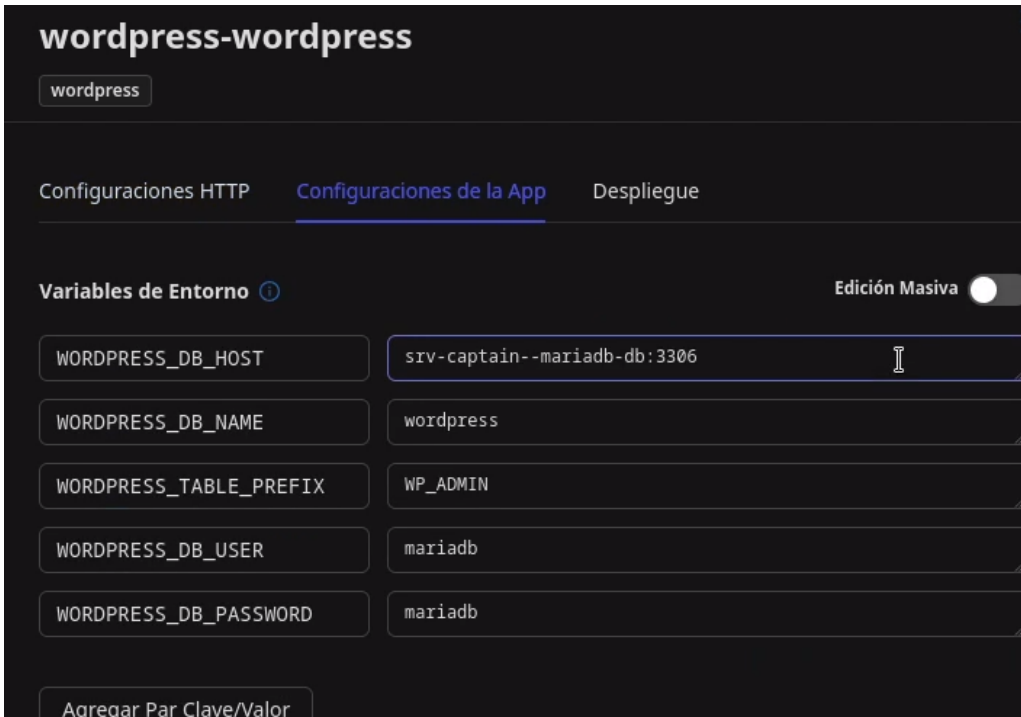
`mariadb`

Configuraciones HTTP   Configuraciones de la App   Despliegue

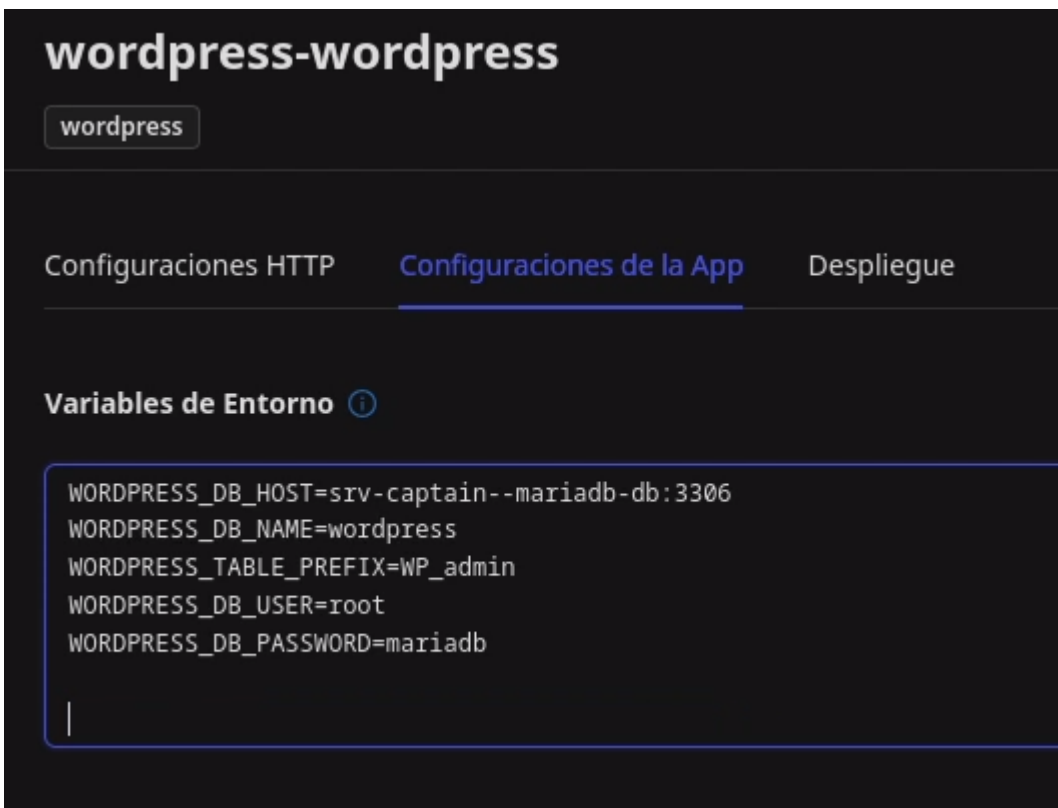
Variables de Entorno ⓘ

|                                  |                      |
|----------------------------------|----------------------|
| <code>MYSQL_ROOT_PASSWORD</code> | <code>mariadb</code> |
|----------------------------------|----------------------|

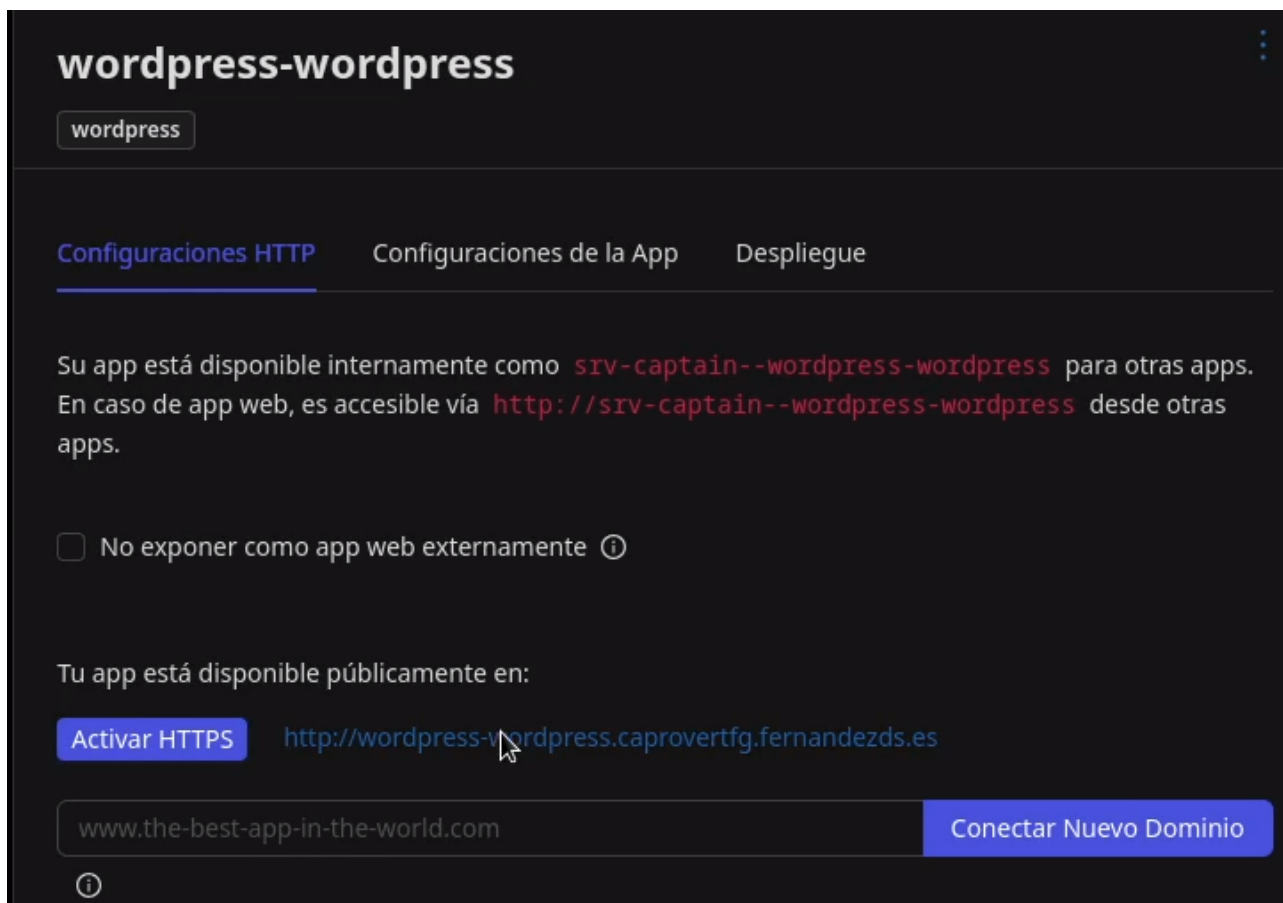
Estas son las variables con la que se ha configurado, que son precisamente las que hay que modificar. Para ello hay que **marcar la opción de edición masiva**, que permite introducirlas como texto directamente.



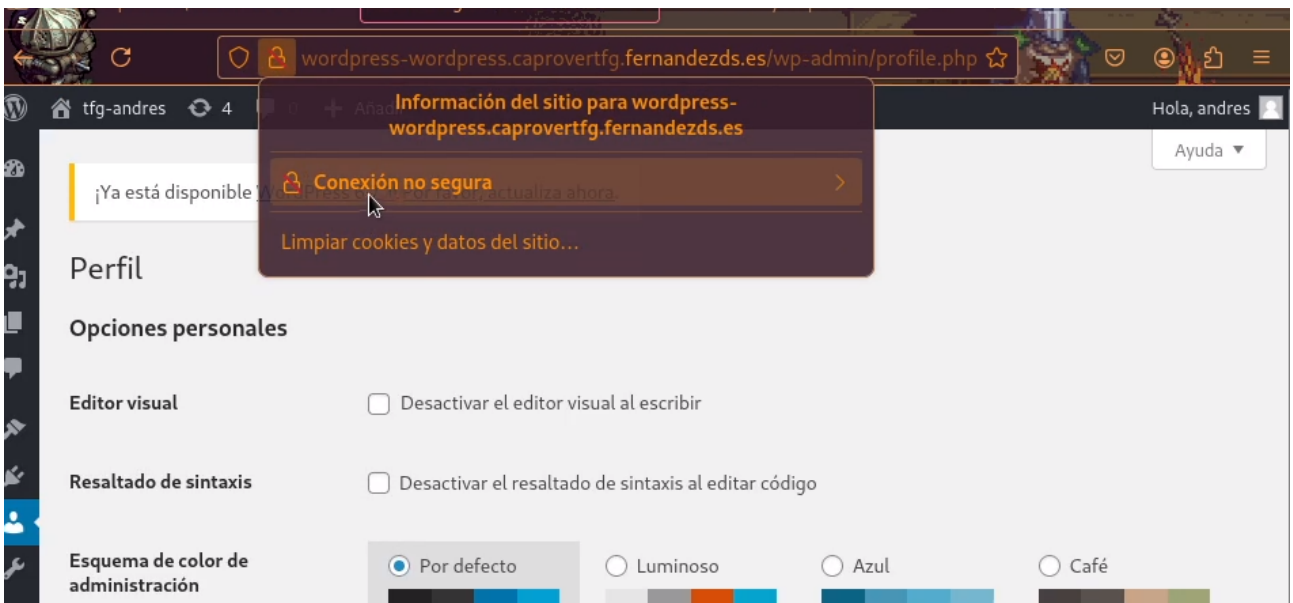
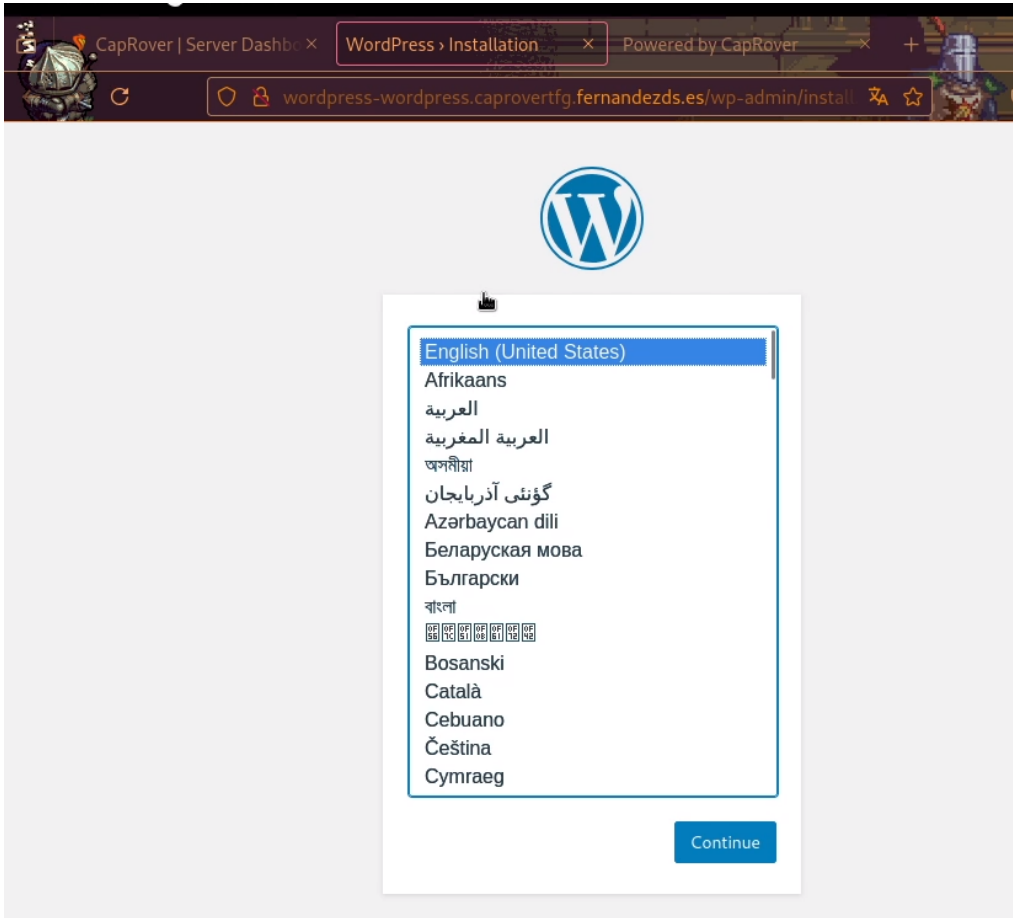
**Quedando de la siguiente manera**, especificando como host, la aplicación de MariaDB antes desplegada, junto con el usuario de la base de datos de mariadb y su contraseña.



Una vez **guardados los cambios**, si pruebo a abrir la aplicación de Wordpress, ya conecta con la base de datos, por lo que **ya funciona correctamente**.

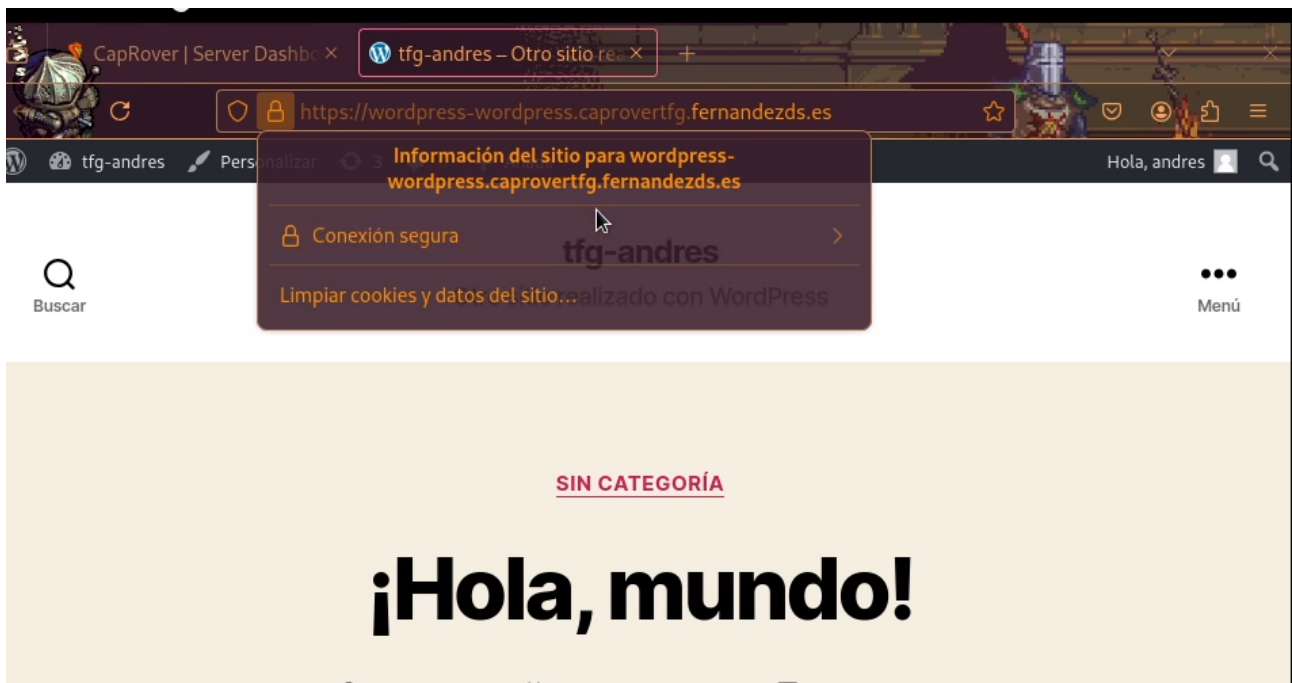
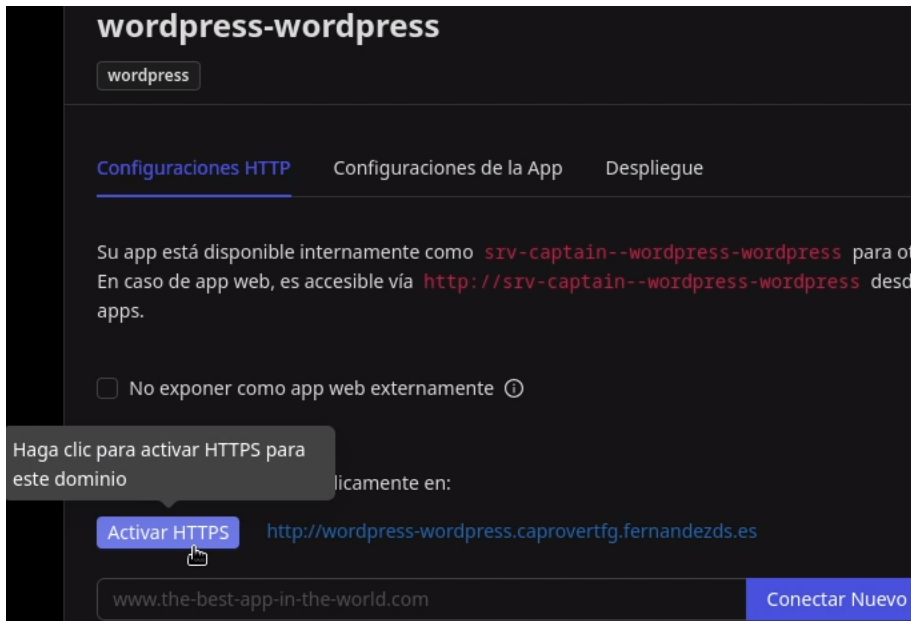


Después de esto, **ya se puede proceder como cualquier otra instalación de Wordpress**.



Andrés Fernández de Santaella

Y por último, **solo bastaría con activar el modo HTTPS**, para dejar finalizado el despliegue de la aplicación.



Y con esto, **el despliegue mediante interconexiones de aplicaciones ha terminado**. Una aclaración final, es que este proceso gracias a **la otra versión de Wordpress del principio**, la que

si contiene base de datos, **no sería necesario**, ya que desplegaría todos los contenedores incluyendo el de la base de datos.

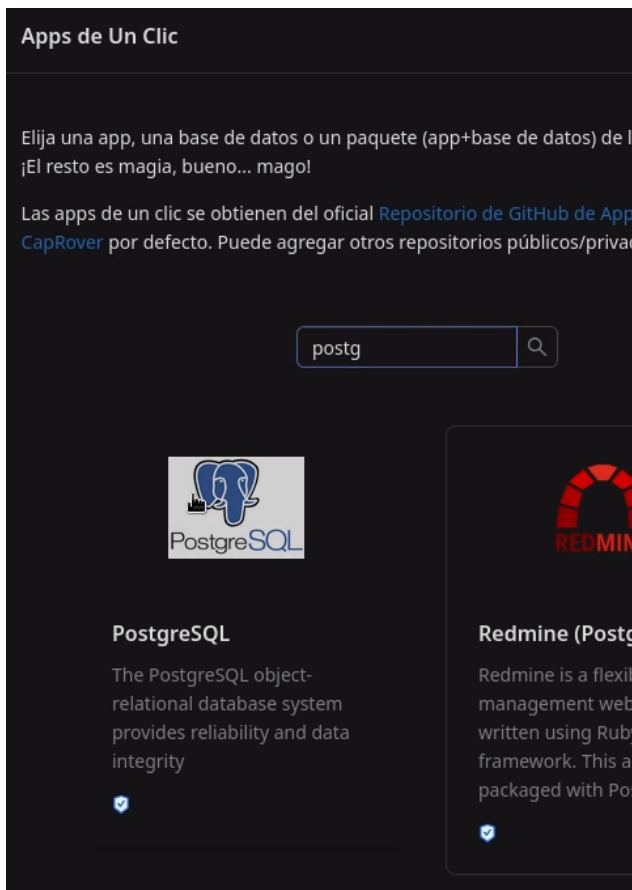
### 6.3 Despliegue Mixto: Django.

Para la demostración de esta parte, **la voy a realizar de manera mixta**, es decir, utilizando las dos tecnologías que he hablado antes. Por una parte voy a utilizar una **base de datos PostgreSQL del listado de aplicaciones de un clic**, junto con la **aplicación Django alojada localmente**. Para ello, cuento ya con la aplicación ya descargada en mi equipo cliente.

Para empezar, en mi archivo captain-definition se encuentra el siguiente contenido:

```
{
  "schemaVersion": 2,
  "dockerfileLines": [
    "FROM python:3.9-alpine",
    "RUN apk update && apk upgrade && apk add --no-cache make g++ bash
git openssh postgresql-dev curl ca-certificates",
    "RUN update-ca-certificates",
    "RUN mkdir -p /usr/src/app",
    "WORKDIR /usr/src/app",
    "RUN pip install --upgrade pip setuptools",
    "COPY ./django_project/ /usr/src/app",
    "RUN pip install -r requirements.txt",
    "COPY ./utils/ /usr/src/utils",
    "EXPOSE 80",
    "CMD sh /usr/src/utils/run.sh"
  ]
}
```

Con la aplicación definida, el primer paso es **crear la base de datos, para este caso Postgresql**, directamente desde el **listado de aplicaciones**.



Como los demás casos, **se especifican las variables y se despliega la aplicación**. Para este caso, **la base de datos se llama “demo”**.

**App Name**  
This is your app name. Pick a name such as my-first-1-click-app

**Version**  
Check out their Docker page for the valid tags <https://hub.docker.com/r/library/postgres>

**Username**

**Password**

**Default Database**

**Deploying your postgres**

This process takes a few minutes to complete. DO NOT refresh this page and DO NOT navigate away!!!

**Progress:**

- ✓ Parsing the template
- ✓ Registering demo
- ✓ Configuring demo (volumes, ports, environmental variables)
- ✓ Deploying demo (might take up to a minute)

```
Postgres is deployed and available as srv-captain--demo:5432 to other apps.  
For example with Node.js: const client = new Client({ user: 'postgres', host: 'srv-captain--demo', database: 'postgres', password: 'postgres', port: 5432})
```

[Finish](#)

El nombre resultante es **srv-captain—demo:5432**. Esto es importante porque **hay que indicárselo luego a Django** para que pueda conectar con la base de datos una vez desplegado.

**Tus Apps**

**Proyectos**

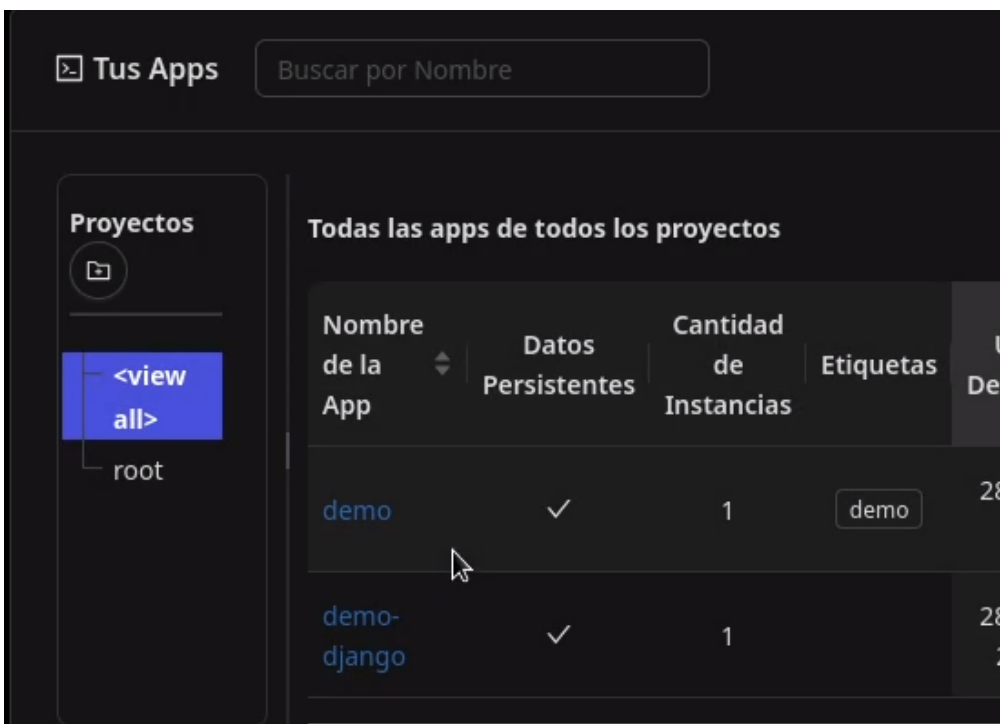
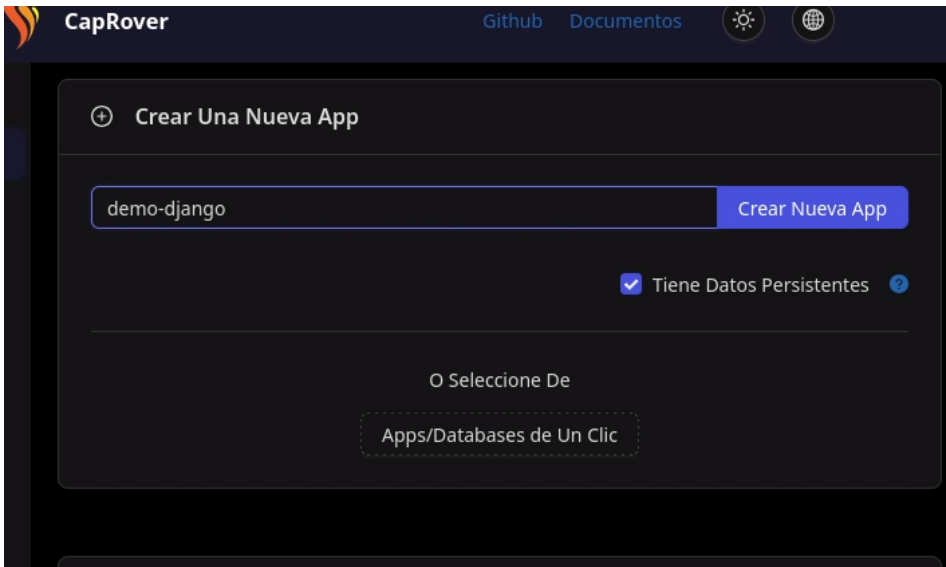
- [<view all>](#)
- root

**Todas las apps de todos los proyectos**

| Nombre de la App | Datos Persistentes | Cantidad de Instancias | Etiquetas | Último Despliegue |
|------------------|--------------------|------------------------|-----------|-------------------|
| demo             | ✓                  | 1                      | demo      | 28/11/20          |



Para poder **desplegar Django**, llamada **“demo-django”** lo primero es **crear el proyecto** y marcando la opción de que **tiene datos persistentes**.



El siguiente paso es **indicarle a Django, las variables de entorno** necesarias para que funcione. En este caso son las siguientes:

```
CAPROVER=True
```

```
CR_SECRET_KEY=$+#q=3(_z3e$9$)6w!wvubk%=2@@9q^b&5v18*-w6q*$nijy$c
```

```
CR_HOSTS=demo-django.caprovertfg.fernandezds.es
```

```
CR_DB_NAME=postgres
```

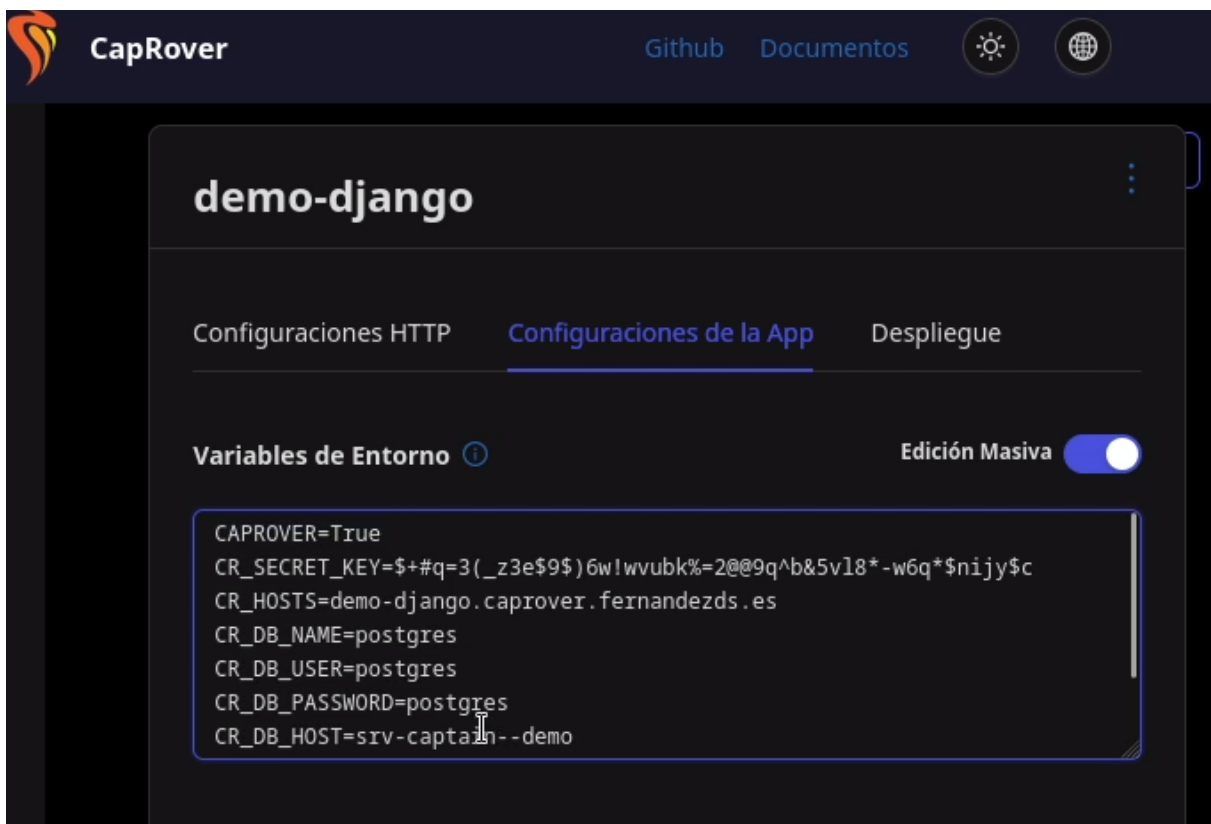
```
CR_DB_USER=postgres
```

```
CR_DB_PASSWORD=postgres
```

```
CR_DB_HOST=srv-captain--demo
```

```
CR_DB_PORT=5432
```

Donde **DB\_HOST**, es la aplicación de **Postgresql para la base de datos** creada antes. (La variable **CR\_HOSTS no incluye la parte de tfg en el dominio** porque se realizó antes de la reestructuración de la máquina que tuve que realizar para que funcionase, pero es lo mismo)



Con las **variables ya definidas en el proyecto**, ahora sí es el momento de **desplegar la aplicación que se encuentra alojada localmente** en mi equipo. Para desplegarla es como los casos anteriores:

```
caprover deploy
```

```
...
```

```
? select the CapRover machine name you want to deploy to: captain-01
```

```
Ensuring authentication...
```

```
? select the app name you want to deploy to: demo-django
```

```
? git branch name to be deployed: master
```

```
? note that uncommitted and gitignored files (if any) will not be pushed to server! Are you sure you want to deploy? Yes
```

```
...
```

```
Build started for demo-django
```

```
...
```

```
Successfully built f8218c11ed16
```

```
Successfully tagged img-captain-demo-django:latest
```

```
Build has finished successfully!
```

```
...
```

```
Deployed successfully demo-django
```

```
App is available at http://demo-django.caprover.fernandezds.es
```

Para no repetirme demasiado, **el proceso es el mismo**. Se **especifica la máquina** captain-01, **el proyecto** que acabo de configurar demo-django, **la rama original** de la aplicación, y que si se quiere **seguir con el despliegue ignorando los archivos sin commit, o en el gitignore**.

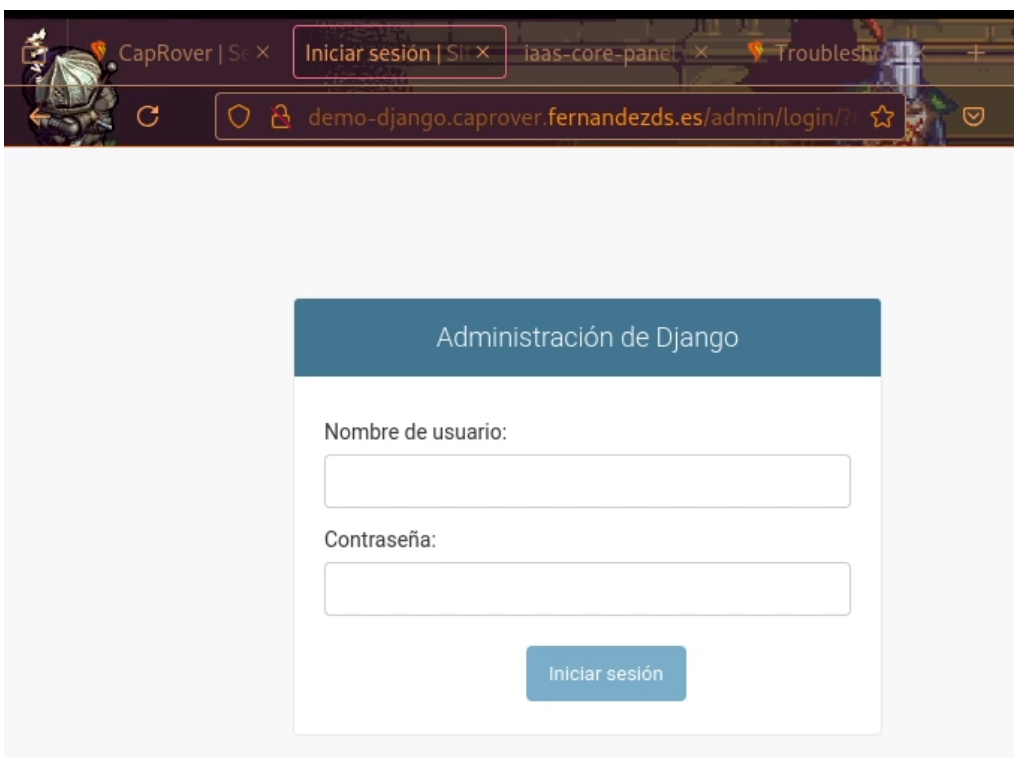
Cuando finaliza, **despliega directamente la aplicación con su respectivo dominio** como los casos anteriores. **La aplicación Django es la original básica sin funcionalidades extras**, es solamente a modo de ejemplo para el despliegue de la misma.



Como se ve, **funciona perfectamente, cargando también la parte de login**. Pero para poder acceder primero, **debo crear un superusuario**. Para ello, entro **directamente al contenedor** de la aplicación y lo creo. En este caso **he creado dos, uno llamado andres, y otro jose domingo**.

Como recordatorio, **para crear un superusuario** es con:

```
python manage.py create_superuser
```



**docker ps**

| CONTAINER ID  | IMAGE         | COMMAND                  | CREATED        |
|---------------|---------------|--------------------------|----------------|
| 335adf15e1f7  | postgres:14.5 | "docker-entrypoint.s..." | 30 minutes ago |
| STATUS        |               | NAMES                    |                |
| Up 30 minutes |               | srv-postgres             |                |

| CONTAINER ID  | IMAGE                     | COMMAND                  | CREATED        |
|---------------|---------------------------|--------------------------|----------------|
| 032902fe4ee3  | img-captain-demo-django:1 | "/bin/sh -c 'sleep 9..." | 30 minutes ago |
| STATUS        |                           | NAMES                    |                |
| Up 30 minutes |                           | srv-demo-django          |                |

...

...

**docker exec -it 032902fe4ee3 sh**

**python manage.py createsuperuser**

**Nombre de usuario (leave blank to use 'root'): andres**

**Dirección de correo electrónico: test@test.com**

**Password:**

**Password (again):**

**Superuser created successfully.**

...

**/usr/src/app #**

**/usr/src/app # python manage.py createsuperuser**

**Nombre de usuario (leave blank to use 'root'): josedomingo**

**Dirección de correo electrónico: test@test.com**

**Password:**

**Password (again):**

**Superuser created successfully.**

...

Andrés Fernández de Santaella

Una vez **creados los usuarios**, ya puedo acceder perfectamente al **panel de control con ambos usuarios**.



Sitio administrativo

| AUTENTICACIÓN Y AUTORIZACIÓN |  |
|------------------------------|--|
| Grupos                       | <a href="#">+ Añadir</a> <a href="#">✎ Modificar</a> |
| Usuarios                     | <a href="#">+ Añadir</a> <a href="#">✎ Modificar</a> |

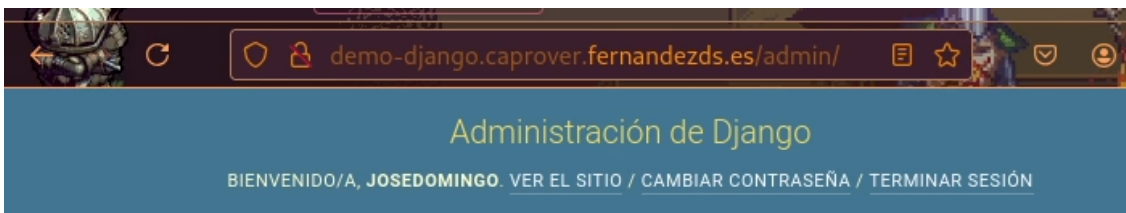
| MY_APP        |  |
|---------------|--|
| Sample models | <a href="#">+ Añadir</a> <a href="#">✎ Modificar</a> |

Acciones recientes

---

Mis acciones

Ninguno disponible



Sitio administrativo

| AUTENTICACIÓN Y AUTORIZACIÓN |  |
|------------------------------|--|
| Grupos                       | <a href="#">+ Añadir</a> <a href="#">✎ Modificar</a> |
| Usuarios                     | <a href="#">+ Añadir</a> <a href="#">✎ Modificar</a> |

| MY_APP        |  |
|---------------|--|
| Sample models | <a href="#">+ Añadir</a> <a href="#">✎ Modificar</a> |

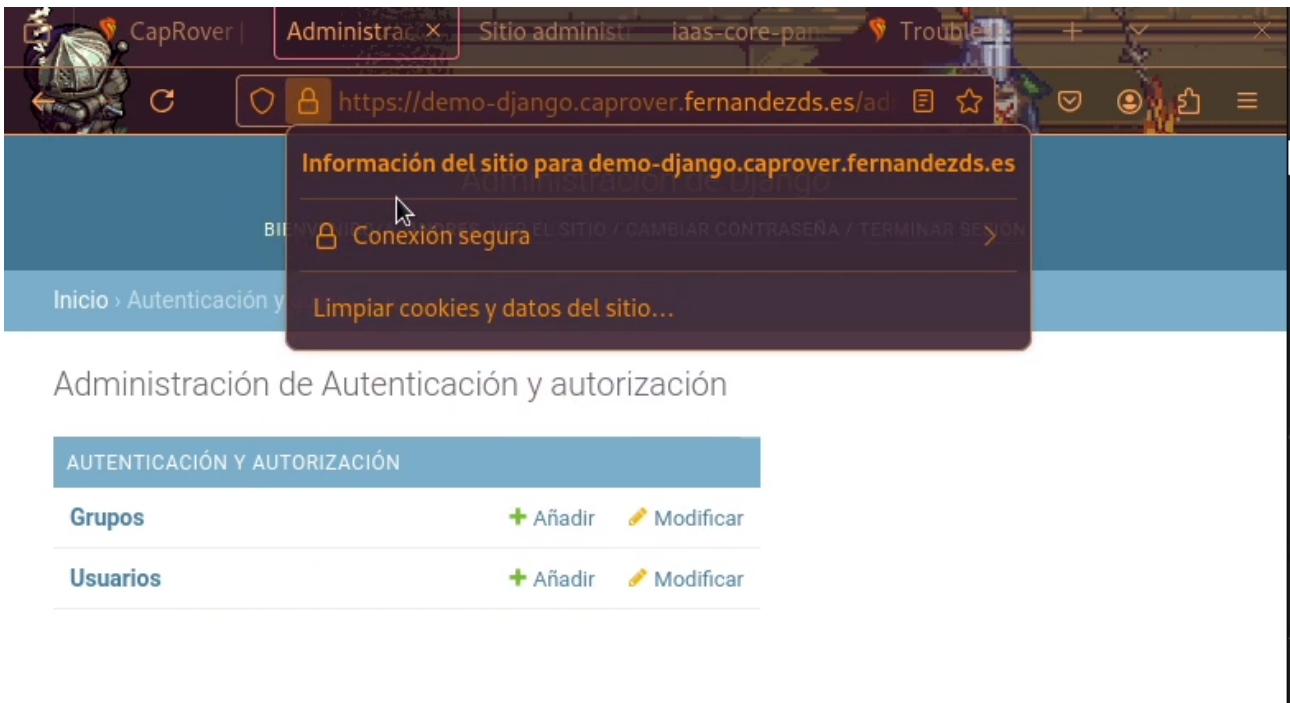
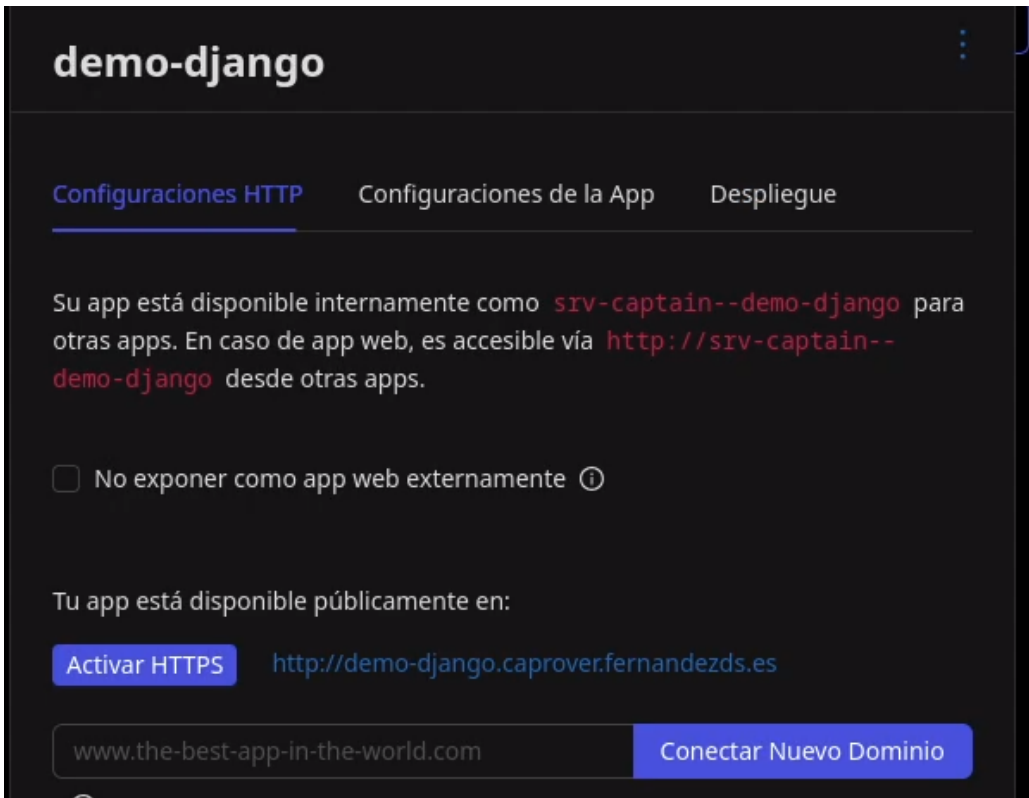
Acciones recientes

---

Mis acciones

Ninguno disponible

Para terminar con el despliegue, como en los demás casos, **solo queda habilitar el modo HTTPS**.



Y con esto **ya estaría desplegada y totalmente funcional la aplicación de Django de manera mixta**. Y para terminar, voy a enseñar en la **máquina servidor como ha creado los servicios** también aparte de los contenedores, para que funcione correctamente.

Estos son los dos últimos, **srv-captain--demo para la base de datos Postgresql**, y **srv-captain--demo-django para la aplicación como tal**.

```
docker service ls
```

| ID           | NAME              | MODE       | REPLICAS |
|--------------|-------------------|------------|----------|
| ljz8v53n1ork | srv-captain--demo | replicated | 1/1      |

```
IMAGE
```

```
postgres:14.5
```

| ID           | NAME                     | MODE       | REPLICAS |
|--------------|--------------------------|------------|----------|
| okypvsn07pd2 | srv-captain--demo-django | replicated | 1/1      |

```
IMAGE
```

```
img-captain-demo-django:1
```

## 7. SOLUCIÓN DE ERRORES

Para esta sección he decidido **compartir los errores que me han ido ocurriendo**, provocados en su mayoría por la falta de recursos de mi máquina, para ser más exacto, de la RAM disponible por lo que he podido averiguar, y el uso del CPU.

### 7.1 Bucle infinito de creación de interfaces de red.

**Este error es debido** a que cuando CapRover realiza un despliegue, este hace **un consumo superior** de lo que hace normalmente como es obvio, y **si la máquina no dispone de memoria suficiente**, la máquina tiende a **comportamientos extraños como este**.

En la **documentación oficial advierten sobre esto, pero no especifican** que puede pasar.



Lo que ocurre tras esto es que **la máquina queda totalmente bloqueada**, procediendo a expulsar al usuario de la conexión ssh e **impidiendo el manejo de la misma** incluso desde la propia terminal del proveedor de la VPS.

Para ser más concreto, **IONOS**, cuando les reporté el problema, **no solo no hicieron nada durante una semana, sino que su única solución era destruir la máquina** junto con todo su contenido, **haya o no copia de seguridad de los datos que contuviese.**

Como es obvio, después de tantos días de espera y sin ayuda de los técnicos, descubrí que desde la propia terminal del proveedor, **logré iniciar sesión entre mensajes de error de creación de interfaces**, y ya al menos **pude restablecer la conexión a internet** para poder acceder por SSH, y **detener inmediatamente los servicios de docker y docker.socket**, que eran los responsables del problema, y **desactivarlos**. Al haber estado activos al inicio, **incluso reiniciando la máquina quedaba inutilizable.**

**La solución:** si tienes pocos recursos asegúrate de **no tener activado al inicio ambos servicios** (unidades de systemd), e ir **controlando con htop** por ejemplo la **ram disponible**, y sobretodo, **libera todo los recursos** que puedas de la máquina, parando servicios innecesarios.

## 7.2 Error de eliminación de CapRover.

Antes de poder averiguar el problema descrito arriba, **tuve que eliminar varias veces Caprover**, pero **para no perder la información hay que conservar el directorio /captain** de la raíz del sistema. En este directorio es **donde se almacenan todos los datos relacionados con caprover**. Si se eliminan los contenedores manualmente, imágenes, servicios, etc **caprover quedará inutilizable**, pero **no permitirá una reinstalación limpia nueva**. Para poder reinstalarlo hay que **seguir la documentación oficial y modificar el registro A** que se creó en los requisitos, por otro distinto, **o no funcionará.**

Según su documentación **esta es la forma correcta de eliminarlo.**

```
docker service rm $(docker service ls -q)
## remove CapRover settings directory
rm -rf /captain
## leave swarm if you don't want it
```

*Andrés Fernández de Santaella*

```
docker swarm leave --force
## full cleanup of docker
docker system prune --all --force
```

Si no se hace de esta manera, como he comentado, **no será posible una reinstalación.**

### 7.3 Recuperar CapRover eliminado accidentalmente.

También me ocurrió que **eliminé uno de los servicios** que sospechaba en su momento que era el **causante del error del bucle infinito de creación de interfaces de red**, y descubrí que incluso reiniciando docker, **si se elimina, CapRover queda totalmente inutilizable.**

**Esta solución solo sirve si no se ha eliminado el directorio /captain** de la máquina servidor. Si por un casual a cualquier persona que siga esta documentación, y decide eliminar alguno de los servicios o se le queda inutilizable Capover, **se puede recuperar de la siguiente manera:**

#### 1. Verificar la existencia del directorio /captain:

```
ls -la /captain
```

#### 2. Verificar el contenido de /captain/data

```
ls -la /captain/data
```

#### 3. Listar los volúmenes Docker (para confirmar si los datos persisten)

```
docker volume ls
```

#### 4. Inspeccionar un volumen específico (como captain--demo-data en mi caso)

```
docker run --rm -v captain--demo-data:/data alpine ls /data
```

#### 5. Eliminar y recrear la red docker\_gwbridge para evitar conflictos

```
docker network rm docker_gwbridge
```

```
docker network create --driver bridge docker_gwbridge
```

## 6. Salir del swarm

```
docker swarm leave --force
```

## 7. Iniciar un nuevo swarm (esto recrea las redes overlay necesarias para CapRover)

```
docker swarm init
```

## 8. Reinstalar CapRover utilizando el directorio /captain existente

```
docker run -p 80:80 -p 443:443 -p 3000:3000 \  
  -e ACCEPTED_TERMS=true \  
  -v /var/run/docker.sock:/var/run/docker.sock \  
  -v /captain:/captain \  
  caprover/caprover:latest
```

Hecho esto, si ha habido suerte, **al entrar de nuevo a la interfaz web**, volverá a permitir el acceso con la contraseña personalizada **manteniendo los datos y los proyectos**.

### 7.4 Recuperar la contraseña de la interfaz web.

Esta contraseña es la que **se define justo al principio**, por defecto es captain42, pero si después de introducir la nueva contraseña **durante la instalación de la CLI**, se olvida **se puede establecer una nueva** de la siguiente manera:

#### 1. Escalar el servicio captain-captain a 0

```
docker service scale captain-captain=0
```

#### 2. Crear un respaldo del archivo de configuración

```
sudo cp /captain/data/config-captain.json /captain/data/config-  
captain.json.backup
```

#### 3. Instalar jq (si no está instalado)

```
sudo apt update  
sudo apt install jq -y
```

#### 4. Eliminar el hash de la contraseña

```
sudo sh -c "jq 'del(.hashedPassword)' /captain/data/config-captain.json > /captain/data/config-captain.json.new"
```

#### 5. Reemplazar el archivo original con el modificado

```
sudo mv /captain/data/config-captain.json.new /captain/data/config-captain.json
```

#### 6. Configurar una contraseña temporal

```
docker service update --env-add DEFAULT_PASSWORD=contraseñanueva captain-captain
```

#### 7. Reactivar el servicio escalándolo nuevamente

```
docker service scale captain-captain=1
```

En la [documentación oficial](#) se pueden encontrar **soluciones para otros errores**.

## 8. VÍDEOS DE DEMOSTRACIÓN

### 8.1 Instalación de CapRover:

[Parte 1.](#)

[Parte 2.](#)

[Actualizar CapRover y crear copia de seguridad rápidamente.](#)

### 8.2 Demostración 1: Aplicación Biblioteca en NodeJS.

[Despliegue De Biblioteca en NodeJs](#)

### 8.3 Demostración 2: Despliegue Automático de Nextcloud.

[Parte 1.](#)

[Parte 2.](#)

### 8.4 Demostración 3: Despliegue Interconectando Aplicaciones: Wordpress

[Parte 1.](#)

[Bonus: Wordpress automatico para que se vea la diferencia entre versiones.](#)

### 8.5 Demostración 4: Despliegue Mixto: Django.

[Despliegue Mixto de Django](#)

## 9. CONCLUSIÓN / AGRADECIMIENTOS

Para concluir con la memoria del proyecto, **me gustaría expresar mi opinión del proceso completo y lo que me ha parecido CapRover.** Como he comentado varias veces, bajo mi opinión es algo increíble, ya no solo por tener la posibilidad de **tener un gran abanico de aplicaciones ya predefinidas** digamos completamente listas para usar, sino también el hecho de **poder tener desplegadas las aplicaciones en producción directamente en internet.**

**Uno de los problemas** durante el curso, o incluso durante la vida profesional de los desarrolladores, estudiantes etc, **es que una vez que se tiene una aplicación,** para poder ponerla **en fase de producción,** por norma general, hay que estar utilizando **herramientas externas** como por ejemplo **Ngrok,** como usamos en alguna practica durante el curso.

**Y hablando de Ngrok** por ejemplo, esta muy bien pero, como haya que realizar **diferentes despliegues** de una aplicación en local, **o durante distintos días** o periodos de tiempo, **es un engorro** el hecho de que cada vez que se utiliza, **este despliega un nuevo enlace,** lo que hace tener

que **modificar el despliegue de la aplicación**, y como se utilice por ejemplo Github Actions, por poner un ejemplo, entre otros servicios, **puede llegar a resultar bastante molesto**.

**Otra alternativa** sería como he mencionado, **utilizar un servicio de terceros como Heroku** por ejemplo, u otros servicios, pero estos son de pago añadiendo un coste extra, o incluso algunos **pueden ser gratuitos pero con funcionalidades muy muy limitadas**.

**Pues con CapRover, se soluciona estos problemas**, por eso es tan genial. Como ya he dicho no solo **soluciona el problema de poder tener todas las aplicaciones** directamente y **sin utilizar** herramientas o **servicios de pago externos** para poder desplegarlas **a producción** en internet, sino que encima de estar **todo centralizado, es totalmente gratuito y de “software libre o abierto”, que es aún mejor**.

**El único inconveniente** que tiene CapRover, es que **se necesitaria tener una máquina VPS, con un dominio**, pero al fin de cuentas, si se van a realizar diversos despliegues al menos, es la mejor opción que se puede elegir, ya que esto, **es un proceso sencillo** el hecho de adquirir una máquina VPS, junto con un dominio y configurarla indicando el FQDN y los registros DNS.

Es un proceso que se puede realizar en una tarde, y una vez realizado, **las ventajas son demasiado extensas en comparación a esta “desventaja”**.

Por último **me gustaría agradecer el trabajo realizado por mis profesores** durante el curso en enseñarnos lo básico de este mundo tan extenso de la administración de sistemas. Personalmente no pude dedicarle el tiempo que se necesitaba o mas bien el que me hubiera gustado para el curso entre problemas personales y familiares.

Una vez fuera de lo que es el **asesoramiento de tener unas clases**, se echa de menos muchas cosas, que en su momento no pude valorar como debía. Por eso mismo si algún estudiante llega alguna vez a leer esto, **aprovechad el tiempo que estéis en las clases**, porque luego creedme que lo vais a echar de menos, tener a alguien que te pueda ayudar, y sobretodo en días en los que todo sale mal y son todo fallos y errores.

Dicho esto, **con esto termino la memoria del proyecto TFG**, podría haber abarcado más contenido acerca de CapRover, pero de ser así, esta memoria sería mucho más larga, de lo que ya es. Un saludo a todos!!!

## **10. BIBLIOGRAFÍA:**

Para la realización del proyecto **me he basado en gran parte en la documentación propia de CapRover**. Aún así dejo algunos enlaces.

[https://es.wikipedia.org/wiki/Computaci%C3%B3n\\_en\\_la\\_nube](https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube)

<https://caprover.com/docs/get-started.html>

<https://cloud.google.com/learn/what-is-paas?hl=es>

<https://github.com/caprover/caprover>

<https://github.com/caprover/caprover/blob/master/LICENSE>