

# **COLEGIO SAGRADA FAMILIA – EL PILAR**



## **PROYECTO FIN DE CICLO**

**“DOCKERIZACIÓN DE UN SERVIDOR COMPLETO EN CLOUD”**

<b>Vº Bº del Director del Proyecto</b>	<b>AUTORES:</b> Gerardo Robustielo Torres, Juan Manuel Nodal Corona  <b>DIRECTORA:</b> María Luisa Suárez Garnacho  <b>CURSO:</b> 2022 / 2023
--	---



# Agradecimientos.

---

Queremos expresar nuestro agradecimiento a los profesores del centro Sagrada Familia – El Pilar por su formación, su paciencia y su trato humano. A Capgemini y los responsables de la FCT, especialmente a Samuel Álvarez Sariego, por permitirnos continuar nuestra formación con sus conocimientos y su experiencia en el sector, que claramente han influido en el desarrollo de este proyecto. Por último, nos gustaría dar las gracias a nuestras familias y amigos por su comprensión, su apoyo y su cariño durante estos duros meses.



# Resumen.

---

El objetivo de este proyecto es la dockerización de un servidor completo que se desplegará en la nube, haciendo uso de una instancia EC2 de AWS. Con ello se pretende simplificar la implementación de un servidor plenamente funcional y ofrecer una solución sencilla de utilizar por parte de las PYMEs.

Se han instalado y configurado los servicios HTTP, FTP, DNS y correo electrónico en una máquina virtual con sistema operativo Ubuntu 22.04 para obtener los ficheros de configuración de dichos servicios y, a partir de éstos, se han creado imágenes Docker de los servicios. Para incrementar la seguridad, se ha utilizado otra instancia EC2 que, a modo de bastión, conecta el servidor con los clientes e Internet. Este bastión contiene un proxy que reenvía al servidor las peticiones que le llegan desde los clientes.

Con esta solución, una PYME puede contar con un servidor plenamente funcional sin tener que lidiar con la complejidad de su instalación y configuración, además de prescindir del coste del hardware y del mantenimiento. A pesar de no estar destinado a ningún cliente concreto, hay que destacar que, al hacer uso de contenedores Docker, se oferta una solución escalable y portátil que puede ser adaptada fácilmente a las necesidades de cualquier tipo de empresa.



# Palabras clave.

---

Plataforma de computación Cloud.

AWS.

Instancia.

Bastión.

Proxy inverso.

Docker.

Contenedor.



# *Abstract.*

---

The main goal of the project is to dockerize a complete server that will be deployed in the cloud using an AWS EC2 instance. Thus, as a result, a simplification of the implementation of a fully operational server, as well as, an easy-to-use solution for SMEs will be achieved.

HTTP, FTP, DNS and email services have been installed and configured in a virtual Ubuntu 22.04 server to obtain the configuration files of said services. From these, Docker images of the services have been created. In order to increase security, another EC2 instance has been used, which as a bastion host, connects the server with the clients and the Internet. The bastion host also works as a reverse proxy, sending the requests from the clients to the server.

Considering this solution, an SME can have a complete functional server without having to deal with the intricacy of its installation and configuration. Despite the fact that no specific client is intended, it should be emphasized that, by using Docker containers, a scalable and portable solution that can be easily adapted to the needs of any type of company is proposed.



## *Keywords.*

---

Cloud computing platform.

AWS.

Instance.

Bastion host.

Reverse proxy.

Docker.

Container.



# Índice general.

---

Capítulo 1. Memoria del Proyecto .....	17
1.1 Objetivos y Alcance del Proyecto.....	17
Capítulo 2. Introducción.....	19
2.1 Estudio de la Situación Actual.....	19
2.2 Evaluación de Alternativas.....	19
2.2.1 Sistema 1 .....	<b>¡Error! Marcador no definido.</b>
2.2.2 Sistema 2 .....	<b>¡Error! Marcador no definido.</b>
Capítulo 3. Aspectos Teóricos .....	23
3.1 Concepto 1 .....	<b>¡Error! Marcador no definido.</b>
3.2 Concepto 2 .....	23
Capítulo 4. Planificación del Proyecto y Resumen de Presupuestos .....	33
4.1 Planificación .....	33
4.2 Resumen del Presupuesto .....	37
Capítulo 5. Análisis .....	38
5.1 Definición del Sistema.....	38
5.2 Requisitos del Sistema .....	39
5.3 Análisis de Interfaces de Usuario.....	<b>¡Error! Marcador no definido.</b>
5.4 Especificación del Plan de Pruebas .....	<b>¡Error! Marcador no definido.</b>
5.4.1 Pruebas Unitarias .....	<b>¡Error! Marcador no definido.</b>
5.4.2 Pruebas de Integración .....	<b>¡Error! Marcador no definido.</b>
5.4.3 Pruebas del sistema .....	<b>¡Error! Marcador no definido.</b>
5.5 Diseño de la Base de Datos.....	<b>¡Error! Marcador no definido.</b>
5.5.1 Descripción del SGBD Usado.....	<b>¡Error! Marcador no definido.</b>
5.5.2 Integración del SGBD en Nuestro Sistema.....	<b>¡Error! Marcador no definido.</b>
5.5.3 Diagrama E-R.....	<b>¡Error! Marcador no definido.</b>
Capítulo 6. Implementación del Sistema.....	43
6.1 Estándares y Normas Seguidos .....	<b>¡Error! Marcador no definido.</b>
6.2 Lenguajes de Programación .....	43
6.3 Herramientas y Programas Usados para el Desarrollo .....	<b>¡Error! Marcador no definido.</b>
6.3.1 Programa 1.....	<b>¡Error! Marcador no definido.</b>
6.3.2 Programa 2.....	<b>¡Error! Marcador no definido.</b>
6.4 Creación del Sistema.....	44

6.4.1	Problemas Encontrados y soluciones adoptadas.....	44
Capítulo 7.	Manuales del Sistema.....	44
7.1	Manual de Instalación .....	47
7.2	Manual de Ejecución .....	47
7.3	Manual de Usuario .....	47
7.4	Manual del Programador.....	47
Capítulo 8.	Conclusiones y Ampliaciones.....	51
8.1	Conclusiones.....	51
8.2	Ampliaciones .....	51
Capítulo 9.	Referencias Bibliográficas.....	53
9.1	Libros y Artículos .....	<b> Error! Marcador no definido.</b>
9.2	Referencias en Internet.....	<b> Error! Marcador no definido.</b>
Capítulo 10.	Apéndices .....	<b> Error! Marcador no definido.</b>
10.1	Contenido Entregado en el pendrive.....	55

# Índice de figuras.

---

<a href="#"><u>FIGURA 4.1. EJEMPLO DE DIAGRAMA DE GANTT .....</u></a>	<a href="#"><u>19</u></a>
<a href="#"><u>FIGURA 4.2. EJEMPLO DE CRONOGRAMA .....</u></a>	<a href="#"><u>20</u></a>
<a href="#"><u>FIGURA 5.6. BOCETO DE UNA INTERFAZ.....</u></a>	<a href="#"><u>26</u></a>



# Capítulo 1. Memoria del proyecto.

## 1.1 Objetivos y alcance del proyecto

El objetivo principal del proyecto es la creación de un servidor en la nube que preste servicios web, de almacenamiento de ficheros, traducción de nombres (DNS) y correo electrónico que pueda ser utilizado por pequeñas y medianas empresas. Para ello, se ha utilizado Docker, que permite el despliegue de aplicaciones previamente creadas y configuradas dentro de contenedores en prácticamente cualquier máquina y entorno. De esta forma, las empresas podrán tener y utilizar un servidor completo y seguro, despreocupándose del coste que conllevan la compra de equipos físicos, la configuración técnica y el mantenimiento.

En este proyecto se aborda un acercamiento y familiarización con los servicios proporcionados por las plataformas Cloud, partiendo de una comparativa entre varias de ellas. El propósito de dicha comparativa es poner de manifiesto cuál de las plataformas ofrece mejores condiciones en su capa de uso gratuita.

Una vez elegida una plataforma Cloud, se crea en ella una infraestructura que contiene una instancia-servidor que proporcionará servicios de alojamiento web, almacenamiento, DNS y correo electrónico a la red de una teórica PYME. Estos servicios no estarán directamente instalados en la instancia, si no que se proporcionarán mediante el despliegue en contenedores de imágenes Docker creadas por nosotros, ahorrando la tediosa instalación y configuración de software. La infraestructura cuenta también con una instancia-bastión que mejorará la seguridad del servidor, controlando el acceso al mismo y reenviando las solicitudes desde los clientes. Este proceso conlleva también la configuración de una serie de elementos de red y seguridad en AWS, como una Gateway de Internet, una Gateway NAT, tablas de enrutamiento y grupos de seguridad. Todos estos elementos serán vistos más adelante.

El proyecto incluye una demostración del funcionamiento del servidor con una muestra de su uso por un potencial cliente.

Finalmente, se subiría el proyecto a un repositorio de GitHub para una futura utilización.



## Capítulo 2. Introducción.

### 2.1 Estudio de la situación actual.

Tras un estudio de lo que nos ofrece el mercado de los servicios informáticos, podemos encontrar una serie de posibles alternativas a nuestro sistema. Si bien algunas de ellas no ofrecen exactamente la misma funcionalidad, merece la pena elaborar una comparación que sirva para poner de manifiesto la útil finalidad de nuestro sistema. Como nuestro sistema requiere del uso de un proveedor de servicios Cloud, el estudio de mercado no se ha limitado sólo a la búsqueda y análisis de alternativas, también se incluye en el [punto 2.3](#) una comparativa de estos proveedores en relación a la calidad de su capa gratuita.

Las alternativas encontradas son las siguientes:

1. Servidor físico.
2. Servidor virtualizado.
3. Zentyal.
4. Zoho Workplace.

A continuación, realizaremos una evaluación de estas alternativas.

### 2.2 Evaluación de alternativas

En este apartado vamos a brindar una descripción de las alternativas previamente mencionadas. Además, se proporcionarán apartados de similitudes con nuestro sistema y de ventajas de nuestro sistema sobre ellas.

#### 2.2.1 Servidor físico.

Máquina física dedicada que se utiliza para ejecutar aplicaciones y brindar servicios a clientes. En ella se puede instalar el mismo software en el que se basan nuestras imágenes Docker.

Similitudes con nuestro sistema:

- Ambos pueden proporcionar exactamente los mismos servicios.

Ventajas de nuestro sistema respecto a esta alternativa:

- Mientras que el servidor físico lleva unos conocimientos técnicos avanzados para configurar los servicios y una importante inversión inicial en hardware, nuestro sistema implica poca o ninguna configuración y ningún gasto en hardware, sólo aquello que nos cobre la plataforma de servicios Cloud.
- Aunque al servidor físico también podrían implementársele medidas de seguridad, el uso del bastión por parte de nuestra infraestructura proporciona una capa adicional de seguridad que oculta el servidor.

- En términos de mantenimiento, un servidor físico requiere la responsabilidad de realizar actualizaciones de hardware, así como el tener que lidiar con problemas técnicos potenciales. Nuestro sistema, cuya parte de hardware está administrada por AWS.

## 2.2.2 Servidor virtualizado.

Servidor dedicado en un entorno de virtualización. Se crea mediante la asignación de recursos de hardware de una máquina física y se ejecuta en un hipervisor, que es el software responsable de gestionar las máquinas virtuales. Proporciona un entorno aislado y completo, con su propio sistema operativo, aplicaciones y recursos asignados

Similitudes con nuestro sistema:

- Ambos pueden proporcionar los mismos servicios.

Ventajas de nuestro sistema respecto a esta alternativa:

- Simplicidad y facilidad de uso: el despliegue de imágenes Docker preconfiguradas ofrece una simplicidad sin par en el proceso de configuración y gestión de los servicios.
- Al utilizar contenedores Docker, nuestro sistema ofrece una mayor eficiencia en el uso de recursos en comparación con la virtualización tradicional. Los contenedores comparten el mismo kernel del sistema operativo, lo que reduce la sobrecarga y permite un despliegue más ligero y rápido.
- Debido a la menor sobrecarga y aislamiento más ligero, nuestro sistema puede ofrecer un mejor rendimiento en comparación con los servidores virtualizados.
- Portabilidad y flexibilidad: nuestro sistema basado en contenedores Docker es altamente portátil y compatible con diferentes plataformas y sistemas operativos. Esto proporciona mayor flexibilidad al permitir el despliegue y ejecución de los servicios en una variedad de entornos.

## 2.2.3 Zentyal.

Distribución de servidor Linux que se centra en proporcionar soluciones integradas para PYMEs. Tiene como objetivo simplificar la administración de infraestructuras de red y proporcionar una solución integral para las necesidades informáticas de la pequeña y mediana empresa.

Similitudes con nuestro sistema:

- Zentyal ofrece los mismos servicios que nuestro sistema.

Ventajas de nuestro sistema frente a esta alternativa:

- Al basarse en imágenes Docker en una instancia de AWS, nuestro sistema proporciona mayor flexibilidad y escalabilidad en comparación con Zentyal, que es una distribución tradicional.

- Las imágenes Docker también brindan a nuestro sistema una portabilidad de la que Zentyal no puede hacer gala. De esta forma, se podría cambiar de proveedor de servicios en la nube o incluso ejecutarlas en local sin problemas de compatibilidad.
- Nuestro sistema da un mayor control sobre la configuración y personalización de los servicios, ya que es posible ajustar parámetros de los contenedores, las versiones de software y las configuraciones de red según las necesidades del cliente.

## 2.2.4 Zoho Workplace.

Suite de productividad en la nube que ofrece una variedad de herramientas y aplicaciones para la actividad empresarial. Es una solución lista para usar que brinda una experiencia unificada y eficiente para la productividad de la empresa.

Similitudes con nuestro sistema:

- Tanto nuestro sistema como Zoho Workplace ofrecen servicios de correo electrónico y otras formas de colaboración, como la compartición de documentos.

Ventajas de nuestro sistema frente a esta alternativa:

- Al estar basado en el uso de imágenes Docker, nuestro sistema ofrece, además de una mayor cantidad de servicios, una personalización y flexibilidad que Zoho Workplace, que es una suite de productividad estandarizada.
- Mientras que la configuración de Zoho Workplace fija unas características de servicio predefinidas, nuestro sistema ofrece un control total de la configuración de los servicios.

## 2.3 Evaluación de alternativas

Como comentamos previamente, para la creación de la infraestructura de nuestro sistema debemos hacer uso de los servicios de un proveedor Cloud. Así, estudiamos las que probablemente sean las más conocidas y usadas plataformas de computación Cloud: AWS, Google Cloud y Azure. Como la intención era llevarlo a cabo utilizando lo máximo posible sus capas gratuitas, se llevó a cabo una comparación de lo que éstas ofrecían, centrándonos en el servicio de despliegue de instancias, ya que será el más utilizado.

Así, tanto AWS como Azure ofrecen 750 horas mensuales de instancias, repartidas entre todas las que se quieran montar, mientras que el servicio de Google nos ofrece solamente una instancia al mes. Quedó así descartado el uso de Google Cloud.

Planteamos entonces una comparativa entre AWS y Azure:

- Los límites del uso de la capa gratuita son más claros. AWS deja muy claro a simple vista hasta dónde llegan sus servicios gratuitos, mientras que Azure usa un sistema de dinero virtual para probar algunos de ellos.
- Mayor cantidad de servicios incluidos. Como en un principio no sabíamos exactamente cuántos servicios del proveedor necesitaríamos, AWS sumó un punto a favor poniendo a disposición del usuario más servicios que Azure.

## Dockerización de un servidor completo en Cloud.

---

- Amplia comunidad. Al ser el servicio más usado a nivel mundial, AWS cuenta con una comunidad más activa, lo cual pone a disposición del usuario más ayuda en caso de que se presenten dudas o problemas.
- Veteranía. AWS es uno de los servicios de computación en nube más antiguos, lo que conlleva una mayor experiencia en términos de funcionalidad y servicio al usuario.

Recabados estos argumentos, quedó claro que AWS era la plataforma Cloud perfecta para llevar a cabo el proyecto y, a la vez, extender nuestro conocimiento sobre Cloud y experimentar con sus servicios sin que ello conlleve elevados gastos.

Algunas ofertas del nivel gratuito de AWS no caducan, estando disponible para todos los clientes. Los servicios más destacables que incluye, además del tiempo de ejecución de instancias ya mencionado, son: 750 horas de uso al mes de servicios de base de datos, 30GB de almacenamiento a distribuir entre las instancias creadas y otros servicios como el Bucket S3, 2 meses de machine learning para científicos de datos y desarrolladores, 15GB de hosting para aplicaciones web y 750 horas de balanceadores de carga, entre otras.

En el siguiente apartado de esta memoria se incluyen todos los servicios de AWS usados para el proyecto.

## Capítulo 3. Aspectos teóricos

En este capítulo vamos a ver una descripción de todos los conceptos, herramientas y tecnologías involucrados en este proyecto, indicando su uso y otros detalles de importancia.

### 3.1 Cloud.

La computación en la nube, también conocida como ‘Cloud computing’ o simplemente Cloud, hace referencia a un modelo de entrega de servicios de computación a través de Internet. En lugar de tener que administrar infraestructuras físicas, los usuarios acceden a recursos informáticos, que puede incluir servidores, almacenamiento, aplicaciones y otros servicios, de forma remota a través de proveedores de servicios en la nube. El fenómeno Cloud ha revolucionado la forma en que se consumen y ofrecen los servicios informáticos, brindando flexibilidad, escalabilidad y eficiencia a las organizaciones.

El concepto de ‘Cloud computing’ se originó en 1960, cuando John McCarthy propuso que algún día la informática sería realizada por empresas de servicios públicos a nivel nacional. La empresa Salesforce se convirtió en 1999 en la primera compañía en ofrecer aplicaciones empresariales a través de Internet. Varios usuarios podrían descargar estas aplicaciones simultáneamente en un navegador a bajo coste. Más tarde, varias empresas, como Google, empezaron a aplicar activamente el concepto de ‘Cloud computing’. El ejemplo más característico es el servicio ‘Google Docs’, que permite trabajar con documentos ofimáticos a través de un navegador.

Las Clouds modernas surgieron en 2006 cuando Amazon, que entonces era una librería online, presentó Amazon Web Services (AWS), lanzando el movimiento del ‘Cloud computing’.

Cloud es uno de los conceptos sobre los que pivota este proyecto, utilizándose manifiestamente al crear toda la infraestructura de red en la nube de AWS, además de usar otros servicios como el Bucket S3.

#### 3.1.1 Proveedores de computación Cloud.

Los proveedores de computación en la nube son las empresas que ofrecen los recursos y servicios que mencionábamos antes. Estos proveedores tienen infraestructuras de servidores y centros de datos distribuidos geográficamente por todo el mundo para alojar y gestionar los recursos. Los proveedores más conocidos y utilizados en este momento son Amazon Web Services (AWS, utilizado para este proyecto), Microsoft Azure, Google Cloud Platform (GPC), IBM Cloud y Alibaba Cloud.

### 3.1.2 Plataformas y servicios de computación Cloud.

Las plataformas de computación en la nube son los entornos virtuales proporcionados por los proveedores Cloud. Permiten a los usuarios acceder y utilizar servicios informáticos sin preocuparse por el hardware que los sostiene. Estos servicios se pueden dividir en tres categorías:

- Infraestructura como servicio (IaaS): Proporciona acceso a recursos informáticos fundamentales, como servidores virtuales, redes y almacenamiento. Los usuarios tienen control total sobre el sistema operativo y las aplicaciones que se ejecutan en la infraestructura.
- Plataforma como servicio (PaaS): Ofrece un entorno de desarrollo y ejecución completo para aplicaciones, que incluye herramientas, bibliotecas y servicios adicionales. Los usuarios pueden desarrollar, implementar y administrar sus aplicaciones sin preocuparse por la infraestructura subyacente.
- Software como servicio (SaaS): Proporciona aplicaciones completas listas para usar a través de la nube. Los usuarios acceden a estas aplicaciones a través de Internet y no necesitan preocuparse por la instalación, mantenimiento o actualización del software.

Las plataformas Cloud se organizan en diferentes regiones. Cada región consiste en uno o más centros de datos físicos, que a su vez pueden contener múltiples zonas de disponibilidad, que son instalaciones separadas pero conectadas dentro de la misma región. La razón principal de tener regiones es brindar una infraestructura de computación en la nube más cercana y accesible a los usuarios finales. Al distribuir los centros de datos en diferentes ubicaciones geográficas, las plataformas Cloud pueden proporcionar servicios y recursos más cercanos a los usuarios finales y a sus aplicaciones, lo que resulta en una menor latencia y tiempos de respuesta más rápidos. Además de la proximidad física, las regiones también se establecen para cumplir con los requisitos de cumplimiento y regulaciones locales (privacidad de datos, protección de la información, etc).

## 3.2 Instancia.

El término instancia hace referencia a un servidor virtual que se ejecuta en la infraestructura de un proveedor de servicios Cloud. En esencia, una instancia es una máquina virtual que ofrece recursos de computación, como CPU, memoria, almacenamiento y red, para permitir la ejecución de aplicaciones y servicios en la nube.

Cuando un usuario solicita una instancia en la plataforma, se crea una copia virtual de un servidor físico dentro del centro de datos del proveedor. Esta instancia puede ser configurada según las necesidades del usuario, incluyendo la elección del sistema operativo, la cantidad de recursos asignados y las aplicaciones y servicios que se desean ejecutar.

Las instancias son altamente escalables y flexibles. Los usuarios pueden solicitar y desplegar instancias adicionales según sea necesario para satisfacer la demanda de sus aplicaciones, y también pueden ajustar los recursos asignados a cada instancia de forma dinámica. Esto permite una gestión eficiente de los recursos, ya que los usuarios solo pagan por los recursos utilizados y pueden adaptar su capacidad de computación de manera rápida y sencilla.

En este proyecto se hace uso de dos instancias EC2 de AWS: una que servirá como base para desplegar las imágenes Docker que proporcionan los servicios y otra que cumplirá la labor de bastión.

### 3.3 Grupos de seguridad.

En el contexto de AWS, los grupos de seguridad son un mecanismo usado para controlar el tráfico de red desde y hacia instancias EC2. Actúa como un firewall virtual, definiendo reglas de seguridad a nivel de puertos, direcciones IP y protocolos.

Cada instancia debe estar asociada como mínimo a un grupo de seguridad, que especifica el tráfico de red permitido y bloqueado. Las reglas de seguridad se dividen en tráfico entrante y tráfico saliente. Así, se puede regular el tráfico para una IP, una red o cualquier origen.

Estos grupos de seguridad siguen el principio de “menos privilegios”, es decir, por defecto bloquean todo el tráfico, teniendo que definir reglas de entrada y salida para permitirlo. Esto proporciona un nivel de seguridad extra al restringir el acceso a las instancias solamente a los recursos y usuarios autorizados.

En el proyecto manearemos dos grupos de seguridad, uno para cada instancia. El grupo del bastión permitirá todo el tráfico saliente y el tráfico entrante para los puertos que utilizan los servicios propuestos que llegue desde la IP pública de la teórica empresa. El grupo del servidor permitirá todo el tráfico de salida y, como el bastión, el tráfico entrante para los puertos que utilizan los servicios que llegue desde la subred en la que se encuentra el bastión.

### 3.4 Pares de claves.

Mecanismo de autenticación utilizado para acceder de forma segura a instancias EC2 y otros servicios de AWS. Constan de dos partes: una clave privada y una clave pública.

Al crear una instancia, se puede asociar un par de claves a dicha instancia. La clave privada se guarda de forma segura en el cliente. La clave pública se almacena en la propia instancia.

El proceso de autenticación funciona de la siguiente manera: cuando se intenta establecer una conexión SSH (Secure Shell) con una instancia EC2, el cliente envía la clave pública correspondiente al par de claves al servidor de la instancia. El servidor verifica si la clave pública coincide con la asociada a la instancia y si es así, permite el acceso. La clave privada, que solo el cliente posee, se utiliza para descifrar la conexión y establecer la comunicación.

El uso de pares de claves proporciona una capa más de seguridad en el acceso a las instancias EC2 en AWS. Al utilizar este método de autenticación, se elimina la necesidad de utilizar contraseñas y se garantiza que quien posee la clave privada puede acceder a la instancia.

En este proyecto se utilizará este método para acceder al desde un cliente al bastión, y desde él otra vez para acceder al servidor.

## 3.5 VPC (Virtual Private Cloud).

Servicio de AWS que permite a los usuarios crear su propia red virtual en la nube. Les proporciona un entorno de red aislado y personalizado en el cual poder tener un control total en todos los apartados: creación de subredes, asignación de direcciones IP, configuración de tablas de enrutamiento, etc. En una VPC se lanzan recursos como instancias, bases de datos o平衡adores de carga.

Más adelante crearemos una VPC que será la base de nuestra infraestructura en la nube.

## 3.6 Subred.

División o segmento de una VPC en el que se pueden lanzar recursos de AWS. Es una red lógica dentro de una VPC y se asocia a una o varias zonas de disponibilidad AWS. Cada subred posee su propio rango de direcciones IP que poder asignar a los recursos que en ella se crean. El concepto es similar a una subred tradicional.

En este proyecto se crean dos subredes: una pública, con salida a Internet, en la que colocaremos la instancia-bastión, y otra privada, sin salida directa a Internet, en la que colocaremos la instancia-servidor.

## 3.7 Tabla de enrutamiento.

Componente clave de una VPC que fija cómo fluye el tráfico de red entre subredes, puertas de enlace y otros servicios en la nube. Contiene reglas que especifican estas rutas que determinan qué destino se alcanza a través de qué puerta de enlace.

Cada VPC tiene asociada una tabla de enrutamiento principal que se crea automáticamente, que además está asociada a las subredes de esa VPC, proporcionando conectividad básica. Además de esta tabla principal, se pueden crear más tablas para establecer configuraciones de enrutamiento específicas, permitiendo controlar de manera más estricta el tráfico de la VPC.

## 3.8 IP elástica.

En AWS, una IP elástica es una dirección IP estática o fija que se asigna a la cuenta AWS que la solicita. Esta IP se puede asociar a una instancia o a otros recursos, impidiendo el cambio de IP cuando se apaguen o reinician estos recursos. A diferencia de las direcciones IP convencionales, las IP elásticas son flexibles, pudiendo asociarse y desasociarse de manera dinámica de los recursos.

En este proyecto se hará uso de dos IPs elásticas: una para la instancia del bastión, que debe tener una IP fija debido a que es a esa IP a la que se le piden los servicios que presta el servidor, y otra a la Gateway NAT, que veremos más adelante.

## 3.9 Puerta de enlace de Internet (Internet Gateway).

Otro de los componentes clave de una VPC. Proporciona conectividad entre la VPC y el Internet público, cumpliendo la función que tendría una clásica puerta de enlace de un router.

Se encarga de que los recursos de la VPC se comuniquen con otros servicios en la nube, sitios web y cualquier otro recurso alojado en Internet, así como de recibir el tráfico entrante de la red de redes.

Es importante señalar que una Internet Gateway sólo proporciona conectividad entre la VPC e Internet. No proporciona conectividad directa a otros servicios de AWS o redes privadas.

Nuestro proyecto hace uso de una de estas puertas de enlace que, asociada a la VPC, proporciona a la infraestructura salida a Internet

## 3.10 Gateway NAT.

Una Gateway NAT (Network Address Translation) es un componente utilizado para permitir que los recursos de una VPC accedan a Internet o a otros servicios externos mientras mantienen una capa adicional de seguridad.

Es útil en situaciones donde se desean mantener los recursos en subredes privadas dentro de una VPC para una mayor seguridad, como es el caso de nuestro servidor. Al utilizar la Gateway NAT, los recursos de la VPC pueden enviar solicitudes salientes a través de él, y el Gateway NAT se encargará de realizar la traducción de direcciones IP necesaria para establecer la comunicación.

Nuestro proyecto utiliza una Gateway NAT que, asociada a la subred privada donde se encuentra el servidor, proporciona conectividad entre éste y el exterior de la subred privada.

## 3.11 Bucket S3.

El servicio S3 de AWS (Simple Storage Service) permite el almacenamiento de datos en la nube. Es altamente escalable y duradero. Permite a los usuarios almacenar y recuperar grandes cantidades de datos de forma segura desde cualquier lugar de internet. Un ‘bucket’ es el contenedor lógico utilizado para dicho almacenamiento de datos.

En este proyecto utilizamos este servicio para almacenar los pares de claves y los archivos de configuración de los servicios y poder disponer de ellos en cualquier momento. Para utilizarlo, se debe instalar en la máquina pertinente la ‘Command Line Interface’ (CLI) de AWS.

## 3.12 Bastión.

También conocido como servidor bastión o servidor de salto, es un servidor seguro que se utiliza como punto de acceso para administrar y acceder a otros servidores situados en una red o entorno protegido. Actúa como puerta de enlace para el acceso remoto a estos servidores y permite a los administradores gestionar de forma segura los recursos de esa red interna.

Este tipo de medida de seguridad reduce la exposición directa del servidor interno, ocultándolo y protegiéndolo de posibles amenazas y ataques externos.

En nuestra infraestructura se crea un bastión y se sitúa en la subred privada de la VPC. Este bastión cumplirá dos funciones: control de acceso al servidor y reenvío de las peticiones de los clientes al servidor (proxy inverso).

## 3.13 Proxy inverso.

Este tipo de proxy actúa como intermediario entre los clientes y los servidores de origen. A diferencia de un proxy normal, que enruta el tráfico desde los clientes hacia el servidor de destino, un proxy inverso enruta el tráfico de los clientes hacia el servidor de origen.

El proxy inverso se coloca en el lado del servidor, en la red del proveedor de servicios. Mejora el rendimiento, la seguridad y la escalabilidad al proporcionar una capa adicional de control y protección.

En nuestra infraestructura, el proxy inverso va instalado en el bastión y reenvía las peticiones de servicios de los clientes al servidor. Hemos utilizado Nginx que, además de ser un servidor web, cumple con la funcionalidad de servidor proxy. Más adelante veremos cómo se configura.

## 3.14 Nginx.

Servidor web y proxy inverso de código abierto diseñado para ofrecer alto rendimiento, escalabilidad y eficiencia en la entrega de contenido. Fue creado por Igor Sysoev en 2004 con el objetivo de superar las limitaciones de los servicios web tradicionales y proporcionar una solución ligera y de alto rendimiento.

Inicialmente, se desarrolló para resolver problemas de concurrencia y rendimiento en servidores web con muchos clientes simultáneos. Para los propósitos de este proyecto, lo que nos importa de Nginx es su funcionalidad como servidor de proxy inverso. Así, instalado y configurado en la instancia-bastión, actúa como intermediario entre los clientes y el servidor de la subred privada, enruteando el tráfico desde los primeros hacia el segundo.

## 3.15 Docker: contenedores e imágenes.

**Docker** es una plataforma de código abierto que permite la creación, despliegue y ejecución de aplicaciones en contenedores. Fue creado por Solomon Hykes y su equipo en la empresa dotCloud, que posteriormente se convirtió en Docker Inc. La primera versión de Docker se lanzó en 2013.

La idea detrás de Docker es proporcionar una forma eficiente y portátil de empaquetar una aplicación con todas sus dependencias en un contenedor. Un **contenedor** es una unidad de software que contiene todo lo necesario para ejecutar una aplicación, incluidas las bibliotecas, las herramientas del sistema, el código y el tiempo de ejecución. Estos elementos vienen incluidos en la **imagen** Docker, una plantilla de solo lectura en la que podemos pensar como una instantánea del estado de un sistema en un momento dado. Una vez desplegada la imagen en el contenedor, nos encontramos con un entorno funcional que puede prestar los servicios que se hayan indicado en la imagen. Una imagen se crea a partir de un archivo **Dockerfile**, fichero que contiene las instrucciones para construir la imagen. Estos son las instrucciones usadas en la construcción de las imágenes de este proyecto:

1. FROM: Especifica la imagen base a partir de la cual se construye la imagen. Debe ir siempre el primero.
2. MAINTAINER: Indica la información de contacto del autor de la imagen.
3. RUN: Se utiliza para ejecutar comandos en el shell dentro de la imagen durante el proceso de construcción. Puede ser utilizado para instalar paquetes, ejecutar scripts, configurar el entorno, etc.
4. COPY: Copia archivos o directorios desde donde se encuentra el Dockerfile hacia la imagen.
5. ENV: Establece un valor para una variable de entorno.
6. EXPOSE: Especifica los puertos en los que el contenedor escuchará cuando se ejecute.
7. CMD: Indica el comando predeterminado que se ejecutará cuando se inicie el contenedor a partir de la imagen. Sólo puede haber un CMD en un Dockerfile. Si se especifica más de uno, sólo el último será válido.
8. ENTRYPOINT: Al igual que CMD, este comando se utiliza para especificar el comando que se ejecutará al iniciarse el contenedor. Sin embargo, ENTRYPOINT proporciona una forma de establecer un comando que no se puede anular fácilmente al ejecutar el contenedor.

La utilidad de Docker radica en su capacidad para resolver problemas de compatibilidad y portabilidad al ejecutar aplicaciones en diferentes entornos. Al utilizar características del kernel de Linux, Docker garantiza que las aplicaciones se ejecuten de manera consistente en múltiples sistemas operativos y entornos, sin importar las diferencias de configuración y dependencias. Además, Docker facilita la gestión de recursos al permitir la creación y destrucción rápida de contenedores según las necesidades. Esto hace que sea ideal para aplicaciones distribuidas, escalables y orientadas a microservicios.

Docker es el otro gran pilar de este proyecto junto a Cloud. Más adelante veremos el proceso completo, desde la creación de las imágenes hasta su despliegue en contenedores.

### 3.16 HTTP y apache2.

El Protocolo de Transferencia de Hipertexto (**HTTP**) es un protocolo de comunicación de la capa de aplicación del modelo OSI que se utiliza para la transferencia de datos en Internet. Fue desarrollado por el World Wide Web Consortium (W3C) y la Internet Engineering Task Force (IETF). HTTP permite la comunicación entre cliente, como navegadores web, y servidores web, facilitando la solicitud y entrega de recursos como páginas web, imágenes, videos y otros contenidos a través de Internet. Utiliza el puerto 80/TCP para las conexiones no seguras y el 434/TCP para las seguras (HTTPS).

**Apache2** es un servidor web de código abierto creado por Robert McCool en 1995 y posteriormente desarrollado por un grupo de voluntarios conocido como Apache Group. Actualmente, el proyecto es mantenido por la Apache Software Foundation, una organización sin ánimo de lucro dedicada al desarrollo de software de código abierto. Es conocido por su estabilidad, seguridad y rendimiento, siendo compatible con varios sistemas operativos, incluyendo Windows, Linux y macOS. Ofrece una amplia gama de características y funcionalidades, como soporte para módulos, configuración flexible, autenticación y muchas más. Este será el servidor web utilizado para crear la imagen Docker que, desplegada en un contenedor, brindará alojamiento web en nuestro servidor.

### 3.17 FTP, vsftpd y Filezilla.

El Protocolo de Transferencia de Archivos (**FTP**) es un protocolo de red que opera en la capa de aplicación del modelo OSI cuya función es la transferencia de archivos entre un cliente y un servidor en una red TCP/IP. Fue desarrollado en la década de los 70 y se ha convertido en uno de los protocolos estándar más utilizados para la transferencia de archivos. Utiliza los puertos 20/TCP, 21/TCP y, en el caso de utilizar el modo pasivo, un rango de puertos definidos por el administrador del servidor entre el 1024/TCP y el 65535/TCP.

**Vsftpd** (Very Secure FTP Daemon) es un servidor FTP de código abierto diseñado para ser seguro y eficiente. Fue creado originalmente por Chris Evans en 1997 y ha sido mantenido y desarrollado por un grupo de colaboradores desde entonces. Vsftpd se enfoca en brindar un alto nivel de seguridad, estabilidad y rendimiento, lo que lo hace adecuado para entornos donde la transferencia de archivos es crítica y se requiere un nivel avanzado de control de acceso. Es el servidor FTP elegido para la creación de la imagen Docker que, desplegada en un contenedor, brindará el servicio de almacenamiento de archivos en nuestro servidor.

Para recibir los servicios de vsftpd en el cliente se utilizará **Filezilla**, un cliente de código abierto para la transferencia de archivos de los protocolos FPT, FTPS y SFTP. Fue desarrollado por Tim Kosse y lanzado por primera vez en 2001.

## 3.18 DNS y bind9.

El protocolo **DNS** (Domain Name System) es un protocolo utilizado en Internet para traducir nombres de dominio legibles para los usuarios en direcciones IP numéricas que las computadoras puedan entender. Proporciona un sistema de resolución de nombres jerárquico y distribuido, lo que permite una traducción eficiente. Fue creado en 1983 por Paul Mockapetris y Jon Postel, y ha sido fundamental para el funcionamiento de Internet al permitir la navegación basada en nombres de dominio en lugar de direcciones IP. Este protocolo utiliza los puertos 53 TCP y UDP.

**Bind9** (Berkeley Internet Name Domain version 9) es una implementación del servidor DNS de código abierto y ampliamente utilizado. Fue desarrollado por Internet Systems Consortium (ISC) y es una de las implementaciones más populares y estables del protocolo DNS. Se encarga de proporcionar servicios de resolución de nombres de dominio, tanto para consultas locales como para consultas en Internet. Es altamente configurable y flexible, permitiendo a los administradores personalizar su configuración según sus necesidades. Al utilizar Bind9, los servidores DNS pueden almacenar y gestionar la información de los registros DNS, responder a consultas de resolución de nombres y facilitar la comunicación entre los nombres de dominio y las direcciones IP correspondientes. Además de los puertos 53 TCP y UDP, bind9 puede utilizar los puertos 953 TCP y UDP, que se utilizan para transacciones entre servidores DNS autorizados.

Bind9 es el servidor DNS elegido para la creación de la imagen Docker que, desplegada en un contenedor, brindará el servicio de traducción de nombres en nuestro servidor.

## 3.19 Correo electrónico: SMTP, IMAP, POP3, Postfix, Dovecot y Thunderbird.

El **correo electrónico** es un servicio de comunicación electrónica que permite el intercambio de mensajes y archivos entre usuarios a través de redes de computadoras. Utiliza el protocolo SMTP para el envío y los protocolos IMAP o POP3 para la recepción.

El protocolo **SMTP** (Simple Mail Transfer Protocol) es el protocolo estándar utilizado para el envío de correo electrónico a través de Internet. Funciona en la capa de aplicación y se encarga de la transferencia de mensajes entre servidores de correo. Utiliza el puerto 25/TCP para la comunicación sin cifrado y el puerto 453/TCP o 587/TCP para la comunicación cifrada. Fue desarrollado en los años 80 por Jon Postel y adoptado como un estándar de facto para el intercambio de correo electrónico. Proporciona un mecanismo confiable y eficiente para el envío de mensajes de correo electrónico en la red.

**POP3** (Post Office Protocol version 3) es un protocolo de acceso a mensajes de correo electrónico que permite a los usuarios descargar sus correos desde un servidor remoto a su dispositivo local. Funciona en la capa de aplicación y se utiliza principalmente en la configuración de correo electrónico de tipo “descargar y eliminar”, donde los mensajes se descargan del servidor y se eliminan de forma predeterminada. POP3 utiliza el puerto 110/TCP para la comunicación sin cifrado y el 995/TCP para la comunicación cifrada. Fue desarrollado en 1984 por Marshall Rose como una evolución de versiones anteriores del protocolo POP. Rose trabajó en colaboración con otros expertos en el Grupo de Trabajo de Operaciones de Correo de la IETF para definir el estándar POP3. Desde entonces, POP3 ha sido ampliamente adoptado y ha sido utilizado en numerosos clientes de correo electrónico y servidores.

**IMAP** (Internet Message Access Protocol) es un protocolo de acceso a mensajes de correo electrónico que permite a los usuarios administrar su correo en un servidor remoto. Funciona en la capa de aplicación y se utiliza tanto para recibir mensajes como para sincronizar los cambios realizados en los buzones de correo entre cliente y servidor. Utiliza el puerto 143/TCP para la comunicación sin cifrado y el 993/TCP para la comunicación cifrada. Fue creado en 1986 por Mark Crispin como una alternativa al protocolo POP como parte del programa de correo electrónico Pine de la Universidad de Washington. Desde entonces ha sido estandarizado por la IETF y ha experimentado varias versiones y mejoras a lo largo de los años.

**Postfix** es un software de servidor de correo de código abierto ampliamente utilizado. Fue desarrollado por Wietse Venema en 1997 como una alternativa a Sendmail. Postfix se ejecuta en sistemas operativos UNIX y es conocido por su seguridad, rendimiento y escalabilidad. Utiliza el protocolo SMTP para enviar y recibir mensajes de correo. Utiliza los puertos estándar antes comentados en el protocolo SMTP. Junto a Dovecot, es uno de los servidores elegidos para la creación de la imagen Docker que, desplegada en un contenedor, brindará el servicio de correo en nuestro servidor.

**Dovecot** es un servidor de correo electrónico de código abierto y uno de los servidores IMAP y POP3 más populares. Fue desarrollado por Timo Sirainen en 2002. Dovecot se integra con Postfix y otros sistemas de correo electrónico y proporciona la funcionalidad para almacenar y acceder a los mensajes de correo electrónico en los buzones de los usuarios. Utiliza los puertos 143/TCP para IMAP y 110/TCP para POP3. Junto a Postfix, es uno de los servidores elegidos para la creación de la imagen Docker que, desplegada en un contenedor, proporcionará a los clientes el servicio de correo electrónico.

**Thunderbird** es un cliente de correo electrónico de código abierto desarrollado por Mozilla. Proporciona una interfaz fácil de usar para enviar, recibir y administrar mensajes de correo electrónico. Es compatible con los protocolos POP3, IMAP y SMTP, lo que le permite trabajar con diferentes servidores de correo electrónico. Es el cliente de correo elegido que tendrán instalados los equipos de los trabajadores de la empresa.

# Capítulo 4. Planificación del proyecto y resumen de presupuestos.

## 4.1 Planificación original.

En la planificación original, la fecha de inicio de este proyecto se fijó en el 12 de abril de 2023. Se marcó la fecha de entrega para el 7 de junio del mismo año. Durante el transcurso de este tiempo se fue realizando la documentación de esta memoria. Entre ambas fechas se marcaron una serie de hitos/entregas en periodos de dos semanas que procedo a enumerar a continuación:

1. 26 de abril. En esta primera quincena estaba previsto realizar el estudio de plataformas Cloud, el registro en la plataforma elegida tras el estudio y una primera familiarización con su entorno, crear una instancia e instalar en ella los servicios LDAP, DHCP, y FTP para obtener los ficheros de configuración que emplearíamos para crear la imagen Docker del servidor. En un principio, la idea era crear una sola imagen que desplegara todo el servidor.
2. 9 de mayo. En la segunda quincena, continuaríamos con la instalación de servicios, en este caso HTTP, DNS y correo electrónico, para obtener los ficheros de configuración y que se emplearían para crear sus respectivas imágenes Docker.
3. 24 de mayo. Durante la tercera quincena tendría lugar nuestra familiarización con Docker y la instalación de esta tecnología en la instancia, creando la imagen del servidor.
4. 7 de junio. Durante la última quincena, desplegaríamos la imagen en una nueva instancia, probando su funcionalidad. Hay que señalar que en un principio no contemplamos cómo se conectarían los clientes al servidor, pensando que sería algo sencillo de hacer. Durante esta quincena se terminaría de documentar la memoria y se entregaría el proyecto.

Esta planificación no salió adelante debido a nuestro desconocimiento de gran parte del contenido del proyecto. Al empezar a trabajar nos fuimos encontrando con problemas, además de cambiar el rumbo del proyecto gracias a los conocimientos adquiridos en el uso de AWS y Docker. En la página siguiente se muestra un diagrama de Gannt de esta primera planificación

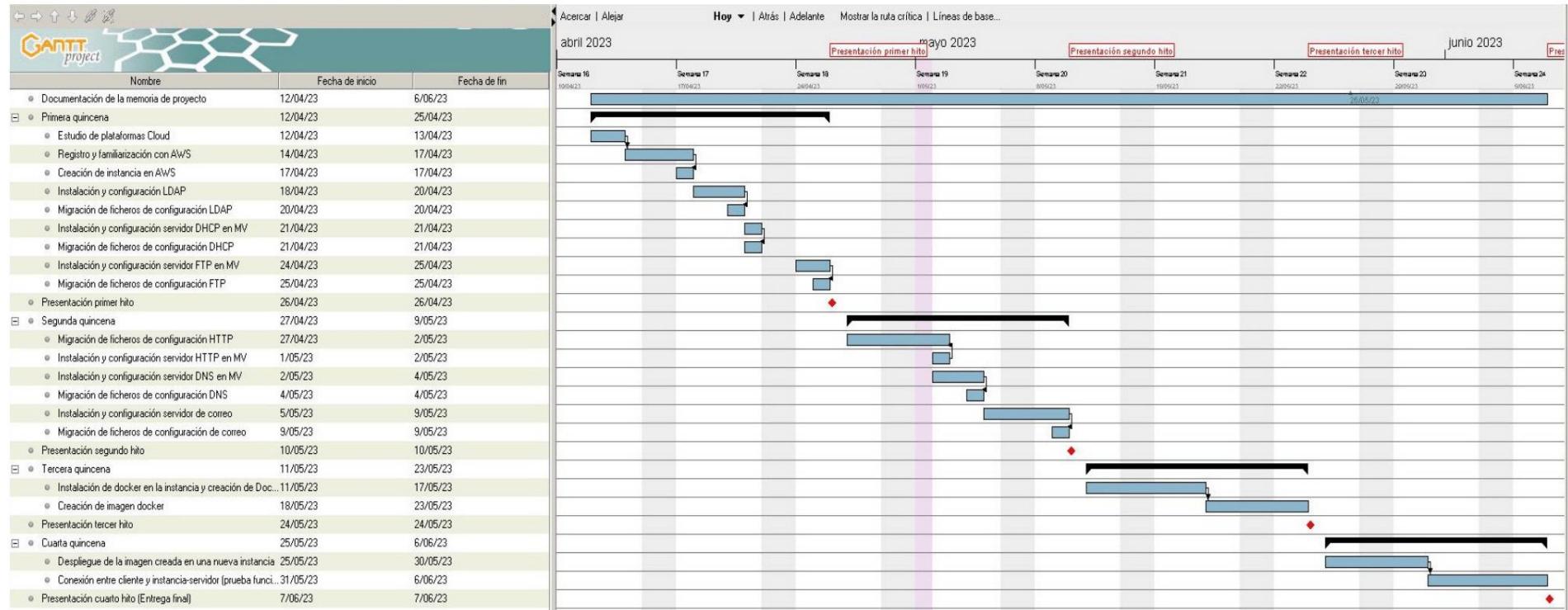


Figura 4.1. Diagrama de la planificación original.

- En azul se marcan las tareas y su duración. Como se puede observar, hay tareas que dependen de otras.
- En negro se marca la duración de las quincenas que separan los hitos.
- ◆ Los rombos rojos marcan la entrega de hitos.

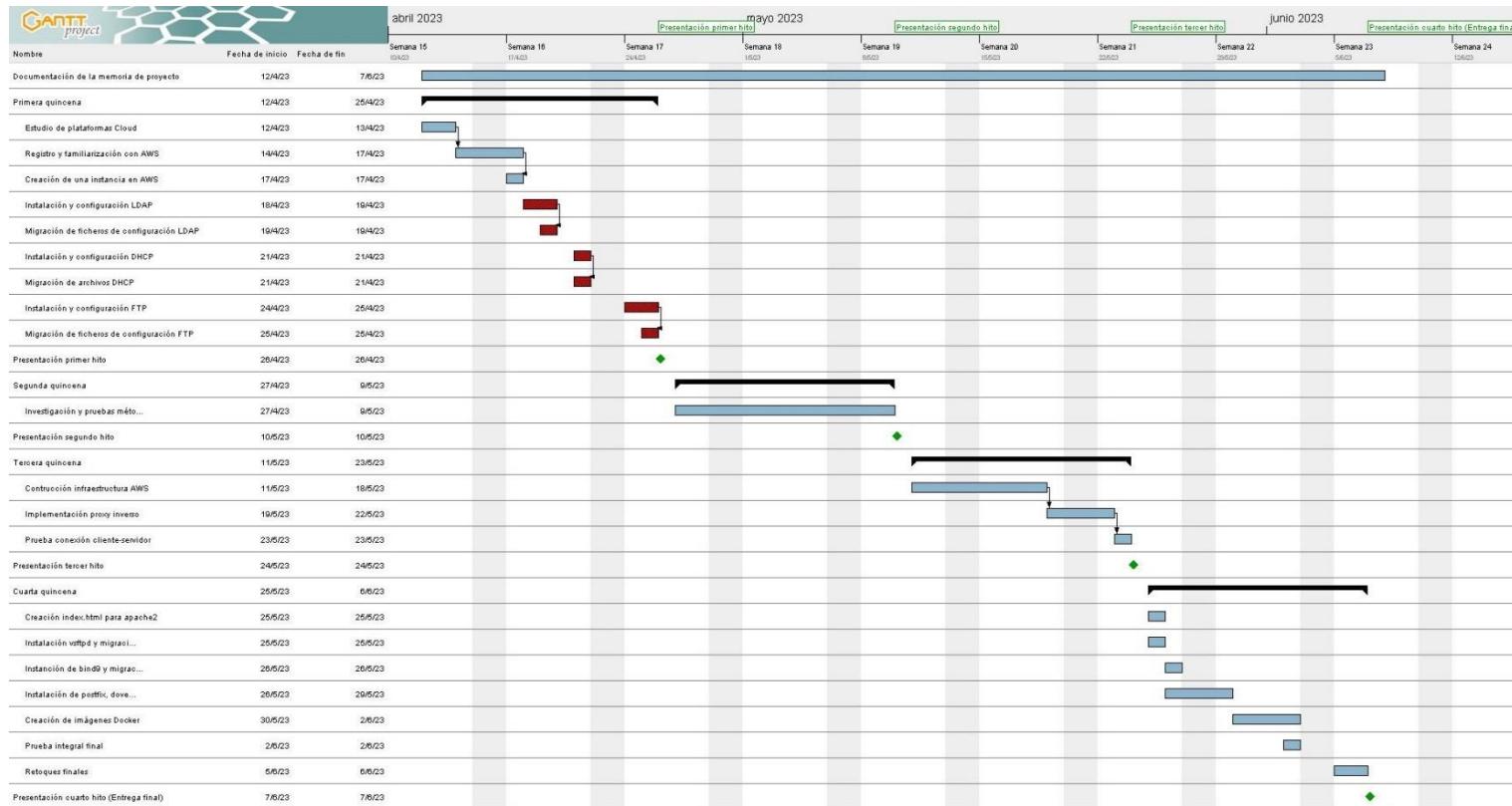
## 4.2 Nueva planificación.

Como se adelantó en el apartado anterior, la planificación original fracasó debido a los problemas encontrados para conectar las instancias de AWS con los clientes locales. Por ello, fue necesaria la creación de una nueva planificación que contemplara lo que habíamos hecho hasta el momento y los pasos futuros. Este cambio de planificación tiene lugar tras la primera quincena, por lo que comparte con la anterior dicha quincena y la tarea de documentación de la memoria. Detallemos esta nueva planificación:

1. 26 de abril. De lo realizado en la primera quincena nos quedamos con el estudio de plataformas Cloud, el registro en la plataforma elegida tras el estudio, la familiarización con el entorno de dicha plataforma y la instancia creada.
2. 9 de mayo. Tiene lugar el cambio de rumbo del proyecto. Se dedicó esta quincena a encontrar la manera de conectar los clientes locales con la instancia de AWS. Se intentaron diferentes métodos, como el túnel VPN o el uso de Zentyal. Estos intentos se relatan en el apartado de '[Problemas encontrados y soluciones aplicadas](#)'.
3. 24 de mayo. Se llegó a la conclusión de que el modo más apropiado y seguro era la utilización de otra instancia a modo de bastión. Durante esta quincena se diseñó y construyó la infraestructura del sistema en AWS, concluyendo con una conexión positiva entre un cliente local y el servidor.
4. 7 de junio. Como en la planificación original, se instalaron los servicios en una máquina virtual Ubuntu 22.04 para obtener sus ficheros de configuración. Sin embargo, para aprovechar mejor las virtudes de flexibilidad y escalabilidad que brinda Docker, se crearon imágenes por cada servicio en lugar de una sola imagen con todos ellos. Se realizó también un manual del registro en Docker Hub, así como de la creación de un repositorio en esta página para almacenar las imágenes creadas. Por último, se realizó una exitosa prueba integral final del sistema y se terminó la documentación de la memoria. Respecto a la planificación original se rechazó el servicio DHCP por incompatibilidades con Docker y el servicio LDAP por falta de tiempo.

En la página siguiente se adjunta el diagrama de Gannt con la nueva planificación.

## Dockerización de un servidor completo en Cloud.



*Figura 4.1. Diagrama de la nueva planificación.*



- En azul se marcan las tareas y su duración. Como se puede observar, hay tareas que dependen de otras.
- En rojo se marcan las tareas canceladas de la primera quincena de la planificación original.
- En negro se marca la duración de las quincenas que separan los hitos.
- Los rombos verdes marcan la entrega de hitos.

## 4.3 Resumen del Presupuesto

Vamos a tomar como referencia para la creación del presupuesto una teórica PYME de 7 trabajadores. El presupuesto se divide en dos partidas. La primera incluye la mano de obra desglosada por conceptos. El precio de nuestra mano de obra será de 33 euros por hora.

Ítem	Concepto	Cantidad	Precio Unitario	Total
0	Formación.	2	300 €	600 €
1	Estudio de la estructura de la empresa.	3	33,00 €	99,00 €
2	Creación de la infraestructura.	3	33,00 €	99,00 €
3	Creación y despliegue de las imágenes Docker.	2	33,00 €	66,00 €
			Subtotal	864,00 €
			IVA (21%)	181,44 €
			<b>TOTAL</b>	<b>1045,44 €</b>

La segunda partida es relativa los costes de los servicios de AWS. Estos costes son mensuales. Estimamos un tráfico de 150GB. Supongamos que las máquinas de la infraestructura no se apagan nunca. Al siempre estar asignadas, las IPs elásticas no cuestan nada. Los precios de AWS ya tienen los impuestos aplicados. Incluimos también un cargo por mantenimiento

Ítem	Concepto	Cantidad	Precio Unitario	Total
0	Tráfico saliente de datos de Amazon EC2 a Internet (1GB).	150	0,09 €	13,5 €
1	IP elástica.	2	0 €	0 €
2	Gateway NAT.	1	36,5 €	36,50
3	Bucket S3 (25GB).	1	0,99 €	0,99 €
4	Instancia EC2 .	1	9,46 €	9,46 €
5	Mantenimiento.	1	50 €	50 €
			<b>TOTAL</b>	<b>109,79 €</b>

# Capítulo 5. Análisis.

## 5.1 Definición del sistema.

Nuestro sistema se sustenta en una infraestructura de red desplegada en AWS. Vamos a enumerar los componentes de dicha infraestructura:

- Una VPC, que cumple la función de la red en la que llevar a cabo el proyecto.
- Dos subredes: una pública, con acceso a Internet, y una privada, aislada del exterior.
- Dos tablas de enrutamiento: una para cada subred, que determinan adónde se dirige el tráfico de red desde la subred o puerta de enlace.
- Dos instancias EC2 t2.micro Ubuntu 22.04: una para el servidor y otra para el bastión. Este tipo de instancias cuenta con un solo GB de RAM. En cuanto al almacenamiento, se destinaron 8GB para el bastión y 19GB para el servidor.
- Dos grupos de seguridad: uno para cada instancia, que funcionan como un firewall virtual que controla el tráfico entrante y saliente.
- Una Internet Gateway asociada a la VPC, que proporciona a la red acceso a Internet.
- Una Gateway NAT, asociada a la red pública, para permitir que el bastión actúe como servidor NAT para proporcionar conexión a Internet al servidor.
- Dos direcciones IP elásticas (fijas): La asignada al bastión (15.236.112.115) y la asignada a la Gateway NAT (13.38.254.170).

Este es el esquema de la infraestructura:

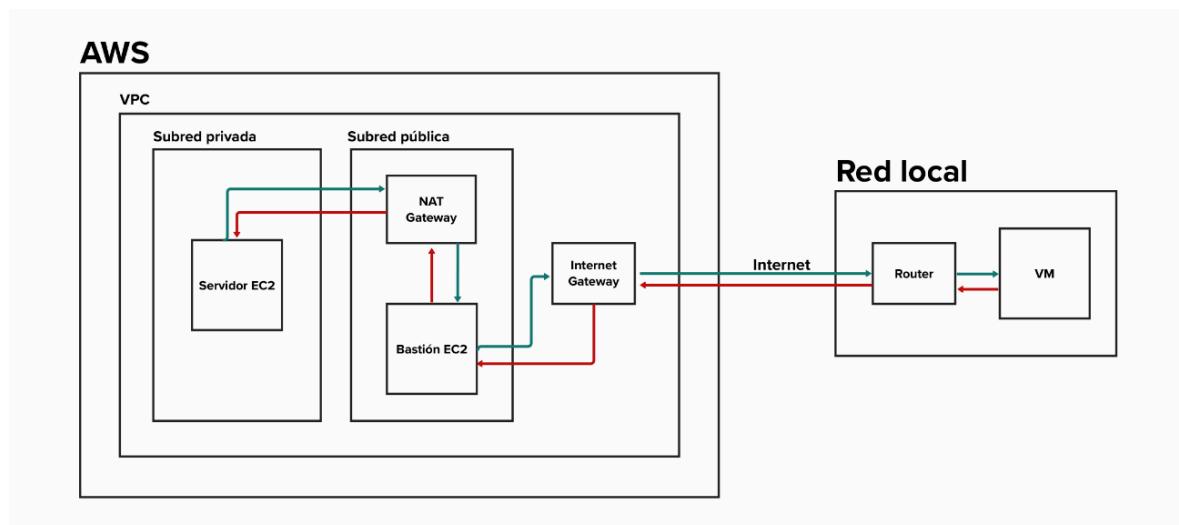


Figura 5.1. Esquema de la infraestructura.

Una vez creada la estructura, se han instalado y configurado los servicios que vamos a dockerizar en un Ubuntu 22.04 virtualizado en nuestra red local. La configuración de los servicios es la misma que tendrán los contenedores que despleguemos al final del proyecto, ya que las imágenes se construirán basándose en estos ficheros de configuración. También se instala Nginx en el bastión y se configuran las reglas del proxy inverso para el reenvío de las peticiones de los clientes al servidor.

Finalmente, se despliegan los servicios dockerizados en contenedores en el servidor, quedando así el sistema listo para su uso.

Los servicios que ofrecerá nuestro sistema son los siguientes:

- Alojamiento web: Se podrá visualizar una página web local de la PYME.
- Almacenamiento de ficheros: Se habilitará el almacenamiento en modo anónimo, permitiendo a los trabajadores de la PYME descargar los archivos subidos al servidor.
- Resolución de nombres: El servidor proporcionará un servicio DNS integral, pudiendo resolver nombres de dominio locales de la propia empresa y nombres de dominio externos a ella.
- Correo electrónico: Los trabajadores de la empresa contarán con un servicio de correo plenamente funcional para comunicarse entre ellos.

## 5.2 Requisitos del Sistema

Se enumeran a continuación los distintos requisitos que deben cumplir las distintas partes involucradas en el proyecto:

- Requisitos de aplicación (R1): Aquellos que debe cumplir el sistema.
- Requisitos de empresa (R2): Aquellos que debe cumplir aquella empresa que quiera implementar nuestro sistema.
- Requisitos de usuario (R3): Aquellos que debe cumplir el usuario del sistema (empleados de la empresa).

En la página siguiente podemos encontrar una tabla que detalla estos requisitos.

Código	Nombre del requisito	Descripción del requisito
R1.1	Visualización de página web local.	Al ingresar en el navegador la IP del bastión o el dominio se debe visualizar la web de la empresa.
R1.2	Descarga de ficheros.	Al introducir la IP del bastión en Filezilla, se debe poder descargar ficheros del servidor.
R1.3	Resolución de nombres.	Al utilizar la IP del bastión como servidor DNS, se deben poder resolver los nombres internos de la empresa y los nombres externos a ella.
R1.4	Correo electrónico.	Mediante el uso de Thunderbird, los usuarios deben poder usar su cuenta de correo y enviar y recibir emails.
R2.1	Registro en AWS.	Si una PYME quiere hacer uso de nuestro sistema, deberá primero completar el registro de una cuenta de usuario en AWS.
R2.2	Medio de pago.	Una vez se decidan a llevar a cabo el registro en AWS, se necesitará un medio de pago para completarlo y poder acceder a los servicios Cloud.
R2.3	Conexión a Internet.	La PYME que implemente la infraestructura debe tener una buena conexión a Internet
R3.1	Conocimientos básicos de informática.	El usuario final del sistema debe saber utilizar las aplicaciones básicas informáticas: navegador, cliente FTP y cliente de correo.

## 5.3 Especificación del plan de pruebas.

Se enumeran a continuación las diferentes pruebas de funcionamiento que se planearon y ejecutaron a lo largo del desarrollo del proyecto para certificar su progreso.

### 5.3.1 Pruebas unitarias.

- Conexión por SSH al bastión desde un cliente en la red local para comprobar que las reglas de los grupos de seguridad funcionan.
- Conexión por SSH al servidor desde el bastión para comprobar que la red interna de la VPC es accesible.
- Instalación de apache2 en el servidor una vez configuradas las tablas de enrutamiento y el proxy inverso para comprobar que hay conexión de servicio con el cliente.
- Instalación y configuración en local de los servicios HTTP, FTP, DNS y correo electrónico y prueba de su funcionamiento previa a la dockerización.

### 5.3.2 Prueba de integración.

- Una vez creadas las imágenes, se realiza una prueba individual de despliegue de cada una de ellas en local para comprobar que la dockerización fue exitosa y que los servicios funcionan.

### 5.3.3 Prueba del sistema.

Prueba final de funcionamiento, desplegando todas las imágenes en el servidor y utilizando un cliente para comprobar que todos los servicios funcionan tal y como se espera de ellos.



# Capítulo 6. Implementación del sistema.

## 6.1 Lenguajes de programación.

Al no ser un proyecto que involucre desarrollo de aplicaciones, no se han visto implicados una gran cantidad de lenguajes. Los que sí están presentes, lo están de una manera muy breve y sencilla. Son los siguientes:

- HTML (Hyper Text Markup Language): Utilizado para crear la estructura y el contenido de páginas web. Utiliza etiquetas o elementos para marcar diferentes partes de un documento y define la estructura lógica y jerárquica del contenido. Cada etiqueta tiene una función específica, representando elementos como párrafos, encabezados, imágenes, enlaces, etc. El fichero ‘index.html’ que contiene el diseño de la página web mostrada por el servicio HTTP está desarrollado en este lenguaje.
- BASH (Bourne Again SHell): Lenguaje de scripting e intérprete de comandos utilizado en sistemas operativos basados en Linux y Unix. Bash proporciona una interfaz de línea de comandos que permite a los usuarios interactuar con el sistema operativo y ejecutar comandos y programas. Además, como lenguaje de scripting, se utiliza para crear scripts para automatizar tareas y procesos. Lo usamos como lenguaje de scripting para crear los pequeños scripts que se incluyen en los Dockerfile para iniciar los servicios y como intérprete de comandos tanto en las instancias EC2 como en los servidores y clientes locales.
- Dockerfile: Se utiliza un lenguaje de sintaxis específico para definir las instrucciones y configuraciones necesarias para construir una imagen Docker. Aunque técnicamente no es un lenguaje de programación completo, se puede considerar como un lenguaje de configuración. Es simple y está basado en instrucciones en texto plano.

## 6.2 Creación del sistema.

### 6.2.1 Problemas encontrados y soluciones adoptadas.

Durante la realización del proyecto nos hemos encontrado con varios problemas. A continuación, se enumeran junto con las soluciones adoptadas o las decisiones tomadas al respecto.

- **Zentyal.** En los primeros pasos del proyecto, cuando nos planteábamos opciones para la configuración de los servicios que daría el servidor, Zentyal fue una de las opciones valoradas. Zentyal es una solución de servidor todo en uno basada en Linux que proporciona una gama de servicios amplia junto a funcionalidades para empresas y organizaciones. Parecía algo apropiado, pero tras instalarlo y empezar a configurarlo nos dimos cuenta de que esta metodología chocaba con el propósito de Dockers de basarse en microservicios escalables y flexibles. Así, se decidió no seguir con Zentyal.
- **Conexión.** La conexión entre el servidor en la nube AWS y un cliente local ha sido claramente la parte del proyecto que mayores problemas ha dado. Al no tener experiencia previa en el sector, la conexión con la nube se antojó una tarea hercúlea, probando diferentes alternativas hasta dar con la solución correcta. Comentemos algunas de las opciones fallidas que se usaron para intentar superar la barrera de la conexión:
  - VPN. Un túnel VPN fue la primera solución que se nos vino a la mente. La creación de una conexión de este tipo era ideal, siendo segura y cifrada, creando un canal privado que compartirían los clientes y el servidor. Se utilizaron dos soluciones distintas de este tipo para intentar lograr la conexión: OpenVPN y los servicios VPN que proporciona AWS, fracasando ambos por la incapacidad de nuestros routers de permitir conexiones de este tipo.
  - PfSense. Distribución de software de código abierto basada en FreeBSD que se utiliza como firewall y enrutador. Proporciona una amplia gama de características y funcionalidades avanzadas para asegurar y controlar el tráfico de red. Al poder cumplir con la función de router, se planteó la idea de que una máquina PfSense podría sustituir a nuestros routers de casa. Sin embargo, tuvimos que desechar esta idea, ya que la conexión a Internet llega a casa mediante cable de fibra óptica y no teníamos un conversor fibra-ethernet ni medios para obtenerlo.
- **Configuración de red en las instancias.** Acostumbrados a la configuración de servidores en local, en los que la configuración de red y otros procesos, como la configuración de un servidor NAT, se realizan modificando ficheros, esta configuración nos dio problemas al empezar a usar las instancias EC2. Fueron varias las instancias que hubo que desechar hasta que nos dimos cuenta de que la configuración de red se realiza exclusivamente en los paneles de AWS. Una vez familiarizados con los servicios Cloud, esto dejó de ser un problema.

- **Bastión-proxy.** La idea del bastión a modo de intermediario fue finalmente la solución perfecta para lograr una conexión segura con el servidor que no lo expusiera directamente a la red. Pero, una vez sobre el papel, nos supuso un nuevo problema. La configuración del enrutamiento ofrecida por AWS mediante las tablas de enrutamiento y los grupos de seguridad no bastaban por si solas para que las peticiones de los clientes llegaran al servidor. La solución fue implementar en el bastión un proxy inverso que reenviara las peticiones. El software elegido para este propósito fue Nginx y funcionó perfectamente.



## Capítulo 7. Manuales del sistema.

En este apartado vamos a presentar todos los manuales de las tareas que se han llevado a cabo a lo largo del proyecto por orden cronológico. Los manuales contienen las instrucciones a seguir para replicar el sistema, así como un manual final en el que se muestra cómo sería un uso por parte del cliente.

### 7.1 Manual de registro en AWS.

Adjuntamos un manual con los pasos a seguir para crear una cuenta en AWS. Este es un requisito fundamental para la realización del proyecto, ya que nos da acceso a todos los servicios Cloud utilizados. Pulsa [aquí](#) para ir al manual.

### 7.2 Manual de creación de infraestructura.

Una vez habilitada la cuenta de AWS, procedemos a crear la infraestructura de red usando los servicios Cloud. Puede parecer un proceso complicado y tedioso, pero esta sensación se debe a la falta de familiarización con los servicios Cloud y el entorno AWS. Una vez formado en el tema, resulta un proceso relativamente sencillo. Pulsa [aquí](#) para ir al manual.

### 7.3 Manual de creación y uso de un Bucket S3.

A lo largo del transcurso del proyecto hemos usado en varias ocasiones un Bucket S3. Éste es un contenedor de almacenamiento en la nube proporcionado por AWS, que se utiliza para almacenar y organizar datos en forma de objetos. Los objetos de un bucket pueden ser archivos de cualquier tipo, como imágenes, vídeos documentos...

La utilidad que nosotros le hemos dado es casi anecdótica, sirviendo para transferir la clave SSH permitente al servidor y para mover el logo de la empresa asociado al html de la web. De todas formas, es interesante conocer un servicio Cloud más. El manual también incluye la instalación del cliente de línea de comandos de AWS y la creación de un usuario con las credenciales para acceder al bucket. Pulsa [aquí](#) para ir al manual.

### 7.4 Manual de instalación del proxy inverso en el bastión.

Una vez creada la infraestructura, había que abordar el tema de cómo iban a llegarle las peticiones de los clientes al servidor, que recordemos está en una red interna. Esto se consiguió mediante la utilización de Nginx, un servidor web y servidor de proxy inverso, que cumplía con todo lo que necesitábamos. Pulsa [aquí](#) para ir al manual.

## 7.5 Manual de registro en Docker Hub.

Una vez configurado todo lo relativo a la parte de AWS, a excepción de los grupos de seguridad, vamos a pasar a lo relacionado con Docker. Empezaremos registrándonos en Docker Hub, donde crearemos un repositorio para subir nuestras imágenes Docker. Pulsa [aquí](#) para ir al manual.

## 7.6 Manual de instalación de Docker en Ubuntu 22.04.

Ahora que ya tenemos una cuenta válida en Docker Hub, vamos a ver cómo se realiza la instalación de Docker en equipos Ubuntu, ya que tanto las instancias como el servidor local se basan en esta distribución. Pulsa [aquí](#) para ir al manual.

## 7.7 Manual HTTP: configuración, imagen Docker, reglas proxy y grupos de seguridad.

Empezamos con los servicios que va a dar el servidor. El primero de ellos será alojamiento local para una web de la empresa. Una vez desarrollado el código de la web, se ha creado la imagen Docker y se ha subido a Docker Hub para su utilización final en nuestro servidor AWS. También veremos la configuración de las reglas del proxy inverso instalado en el bastión y la configuración de los grupos de seguridad para abrir los puertos necesarios. Pulsa [aquí](#) para ir al manual.

## 7.8 Manual FTP: configuración, imagen Docker, reglas proxy y grupos de seguridad.

Continuamos con el manual del servicio FTP. Para obtener el fichero de configuración se ha instalado vsftpd en un Ubuntu 22.04. Una vez en posesión del fichero ya configurado, se ha creado la imagen Docker del servicio, se han añadido las reglas al proxy inverso y se han configurado los grupos de seguridad de las instancias. Pulsa [aquí](#) para ir al manual.

## 7.9 Manual DNS y correo electrónico: configuración, imagen Docker, reglas proxy y grupos de seguridad.

Por último, tenemos los servicios de DNS y correo electrónico. Estos dos servicios, al ir tan íntimamente ligados, se ha decidido que vayan juntos en la misma imagen Docker. El manual incluirá la configuración de los ficheros, la creación de la imagen Docker, las nuevas reglas del proxy y la configuración de los grupos de seguridad. Pulsa [aquí](#) para ir al manual.

## 7.10 Manual de despliegue de imágenes.

Ahora que ya tenemos todas las imágenes en Docker Hub, es hora de desplegarlas en contenedores en nuestro servidor de AWS. Este es el paso final del proyecto antes de la comprobación de usuario. Pulsa [aquí](#) para ir al manual.

## 7.11 Manual de usuario/prueba de funcionamiento.

Este manual que servirá tanto de instrucciones sobre lo que tiene que hacer un usuario para utilizar el sistema como de demostración de funcionamiento, ya que el que un usuario reciba los servicios es la prueba definitiva de que el proyecto ha sido un éxito. Se utilizará un Linux Mint en una red local con acceso a Internet para la prueba. Tiene instalados un navegador, Filezilla y Thunderbird. Pulsa [aquí](#) para ir al manual.

## 7.12 Manual de registro en Git Hub y subida del proyecto a un repositorio.

Lorem ipsum  
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum

Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum  
Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum. Pulsa [aquí](#) para ir al manual.



# Capítulo 8. Conclusiones y ampliaciones.

## 8.1 Conclusiones.

Se ha elaborado una infraestructura en Cloud capaz de dar soporte a un servidor seguro que, haciendo uso de contenedores Docker, da servicios de alojamiento web, almacenamiento de ficheros, traducción de nombres de dominio (DNS) y correo electrónico.

El sistema cumple perfectamente con las exigencias que pudiera tener una pequeña empresa, permitiendo el uso de páginas web locales, descarga de archivos de empresa de un directorio al que pueden acceder los empleados, traducción de nombres de dominio e IPs tanto de la propia red de la empresa como externos y un totalmente funcional sistema de correo electrónico interno. Además, con el uso del bastión y los estrictos grupos de seguridad de AWS, se ha conseguido una seguridad elevada para el servidor.

Dejando a un lado la parte técnica, el proyecto también ha cubierto con creces las expectativas de aprendizaje de los autores en dos tecnologías con mucho presente y futuro, como son los servicios Cloud y Docker.

En resumen, el resultado del proyecto es muy satisfactorio para sus autores, recompensando el gran esfuerzo que han puesto en su desarrollo.

## 8.2 Ampliaciones

Se manejan una serie de ampliaciones que, por falta de tiempo, no se han podido aplicar al sistema. Serían las siguientes:

- Implementación de las versiones seguras de los protocolos de los servicios que proporciona el servidor: uso de HTTPS, SFTP y correo con cifrado. (SMTSP, IMAPD y POP3D)
- Incorporación de más servicios al servidor: como podrían ser una estructura LDAP o mensajería instantánea.
- Mejora del servicio FTP: Crear una estructura de directorios para que cada usuario y que, además de tener un directorio personal, puedan acceder a recursos de su departamento y otros recursos comunes de la empresa.
- Personalización de sistemas: Según el tipo de empresa que fuese el potencial cliente, ponerse de acuerdo de antemano sobre el presupuesto del que disponen y crear un sistema con unas prestaciones mayores para incrementar su rendimiento y mejorar así los servicios prestados.



## Capítulo 9. Referencias bibliográficas

Estas son las fuentes de documentaciones, explicaciones o contenidos de otros tipos que nos han ayudado en el desarrollo del proyecto

Páginas Web consultadas para cualquier aspecto relacionado con el desarrollo del sistema o su documentación.

**Formato sugerido:**

[<PrimerApellidoAutor><DosUltimosDigitosDelAño>]<Apellidos1, Nombre1; Apellidos2, Nombre2;...>. “<Título de la página Web>”. <URL>. <Año en el que se consultó (4 cifras)>.

**Ejemplo:**

**[Redondo07]** Redondo L., J. Manuel; De Tal y Cual, Menganito. “Título de la página Web de ejemplo”. [www.unaurcualquiera.com](http://www.unaurcualquiera.com). 2007.

Si tenemos más datos que permitan localizar la información dentro de la página, podemos ponerla donde consideremos oportuno.

Esta referencia es real (se usa dentro del documento) y debe dejarse aquí siempre que usemos el cuestionario que la menciona en la sección de usabilidad.

**[Hassan08]** Hassan Montero, Y. “Guía de Evaluación Heurística de Sitios Web”. <http://www.nosolousabilidad.com/articulos/heuristica.htm>



## Capítulo 10. Anexos.

### 10.1 Anexo 1: Contenido entregado en la memoria USB.

Debe contener estructura de directorios con la memoria en pdf, programas, presentación, máquinas virtuales utilizadas para la defensa,....

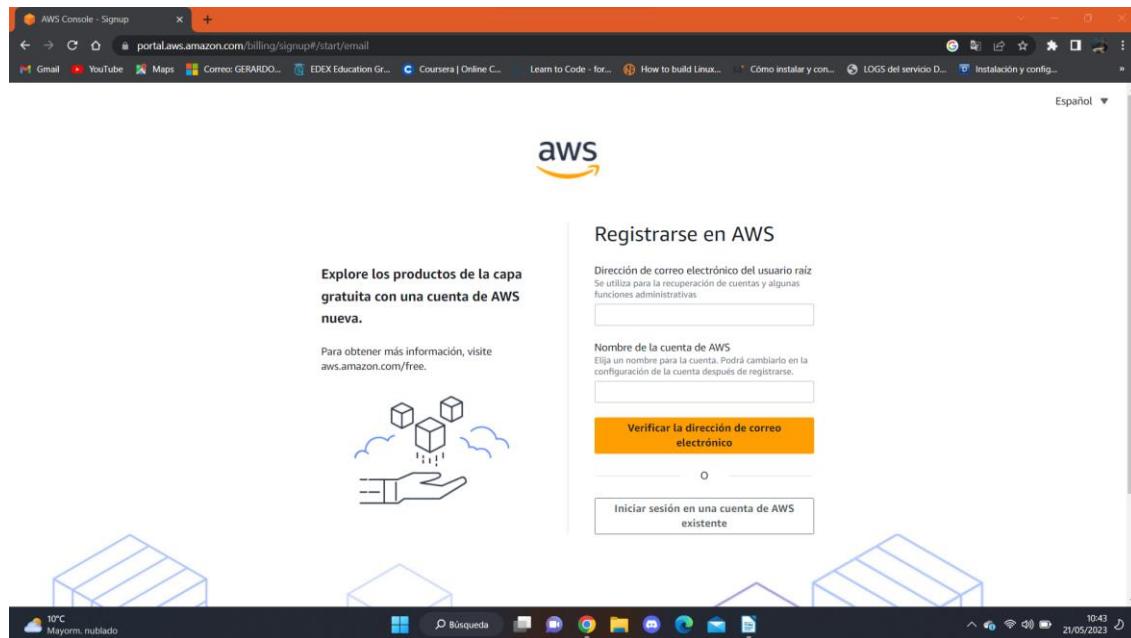
En la unidad de almacenamiento entregada junto al proyecto se puede encontrar el siguiente contenido:

1. Una copia en PDF de este documento de memoria del proyecto.
2. Una copia de la presentación utilizada para la defensa del proyecto.
3. Un archivo ‘mint.ova’ (Open Virtualization Format Archive) que contiene el cliente Linux Mint utilizado en la defensa del proyecto.

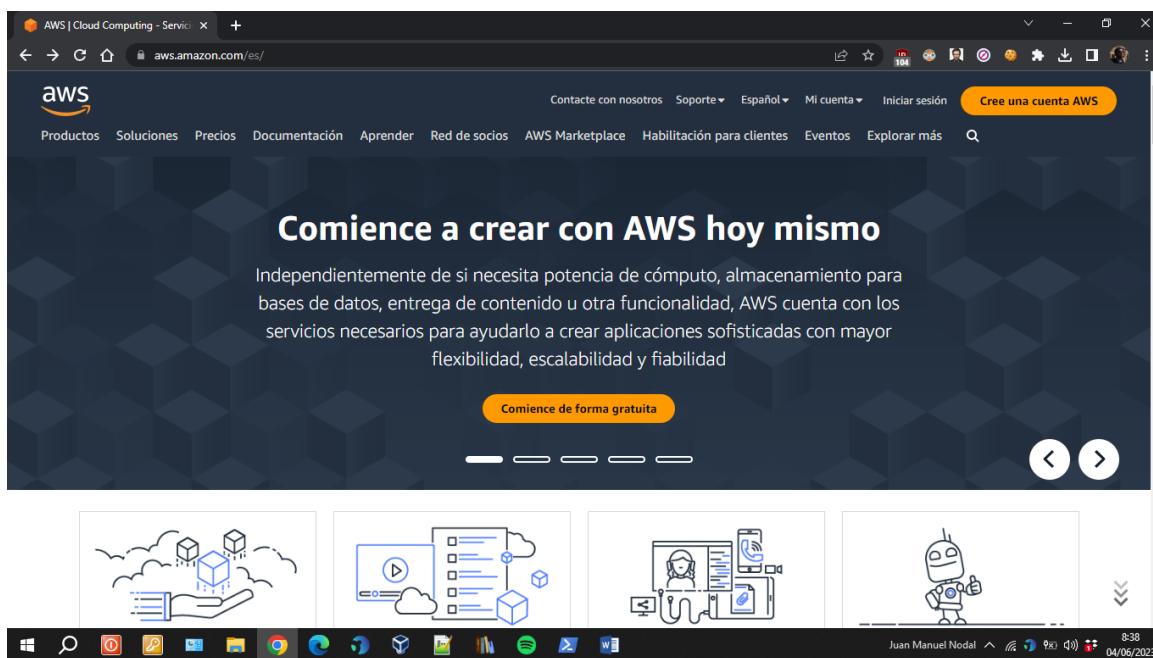
## 10.2 Anexo 2: Manual de registro en AWS.

Para utilizar instancias EC2 y los demás servicios Cloud, es necesario registrar una cuenta en AWS y configurarla. No es un proceso complicado, aunque puede resultar confuso si no se está familiarizado con el tema. Se podría resumir el proceso en los siguientes pasos:

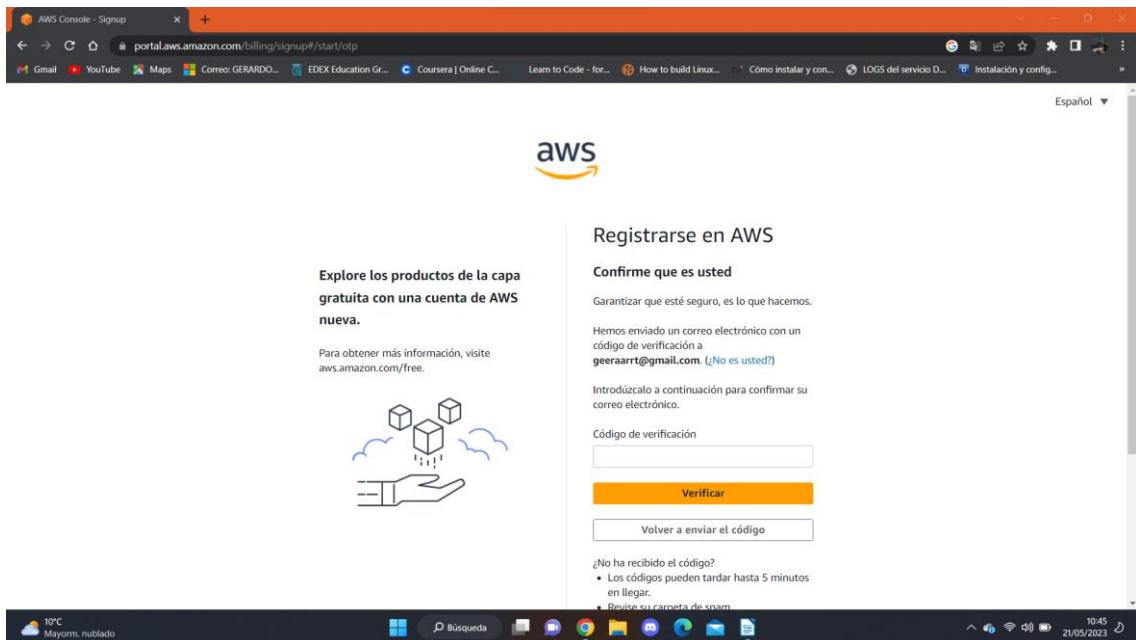
1. Accede a la web de AWS (<https://aws.amazon.com/es>) y haz click en ‘Comience de forma gratuita’.



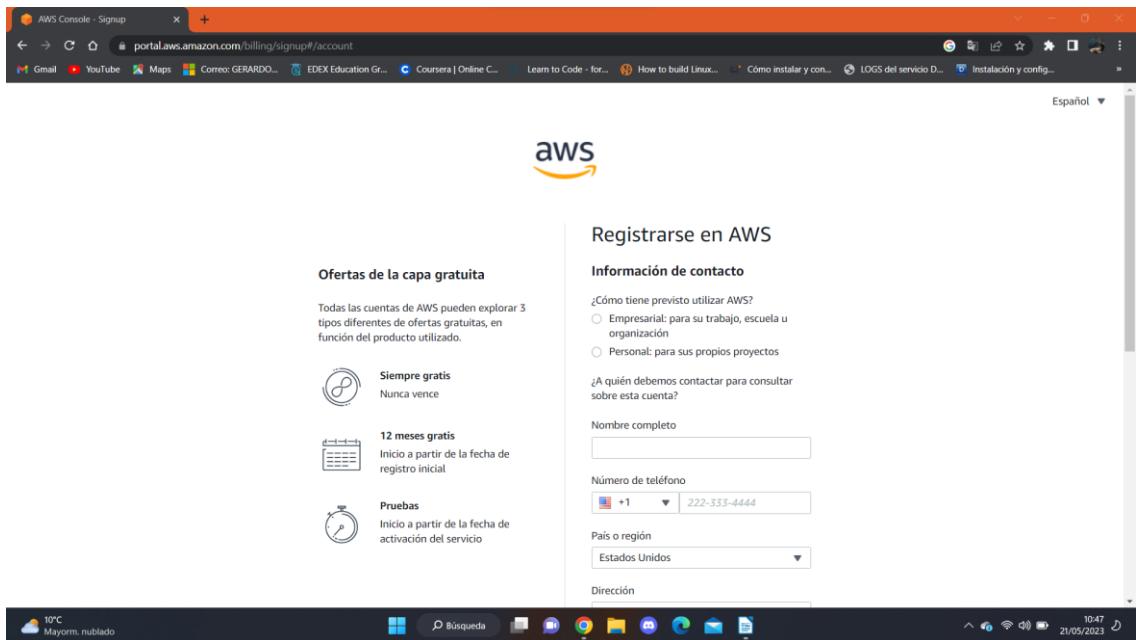
2. Introduce una dirección de correo electrónico válida para el usuario raíz y un nombre de la cuenta de AWS. Debe verificarse accediendo a la cuenta de correo que se facilita mediante un código de verificación que nos envían.



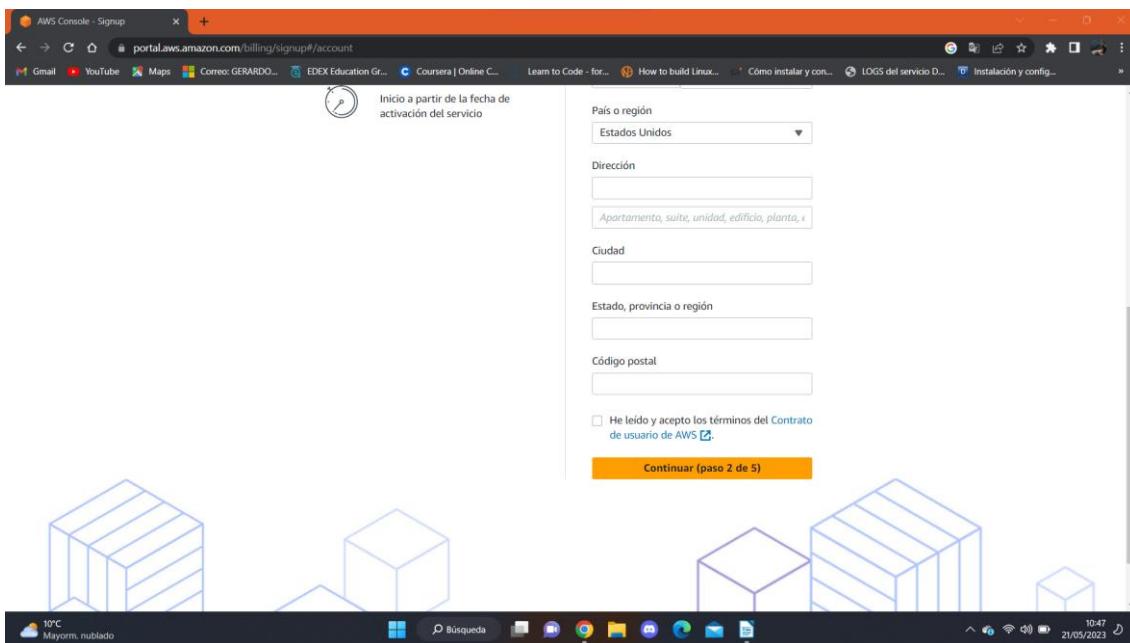
3. Una vez introducido el código, llega el momento de introducir nuestra contraseña.



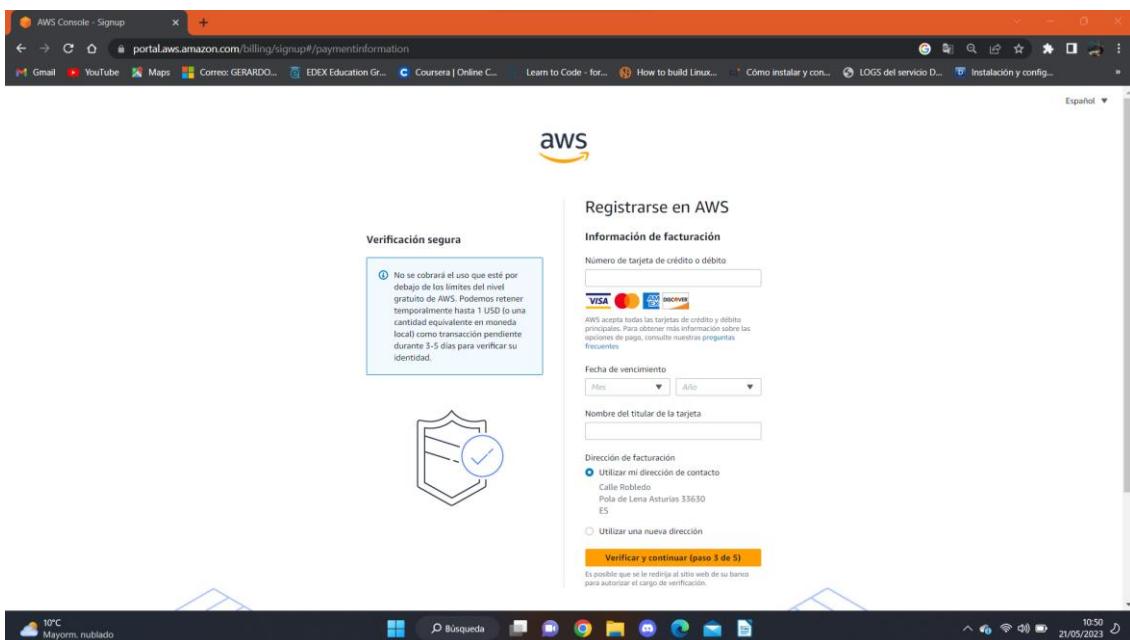
4. A continuación, se piden los datos personales del usuario. También se pregunta si el uso previsto de la cuenta será de tipo empresarial o personal. Debemos asegurarnos de marcar la casilla de los términos del Contrato



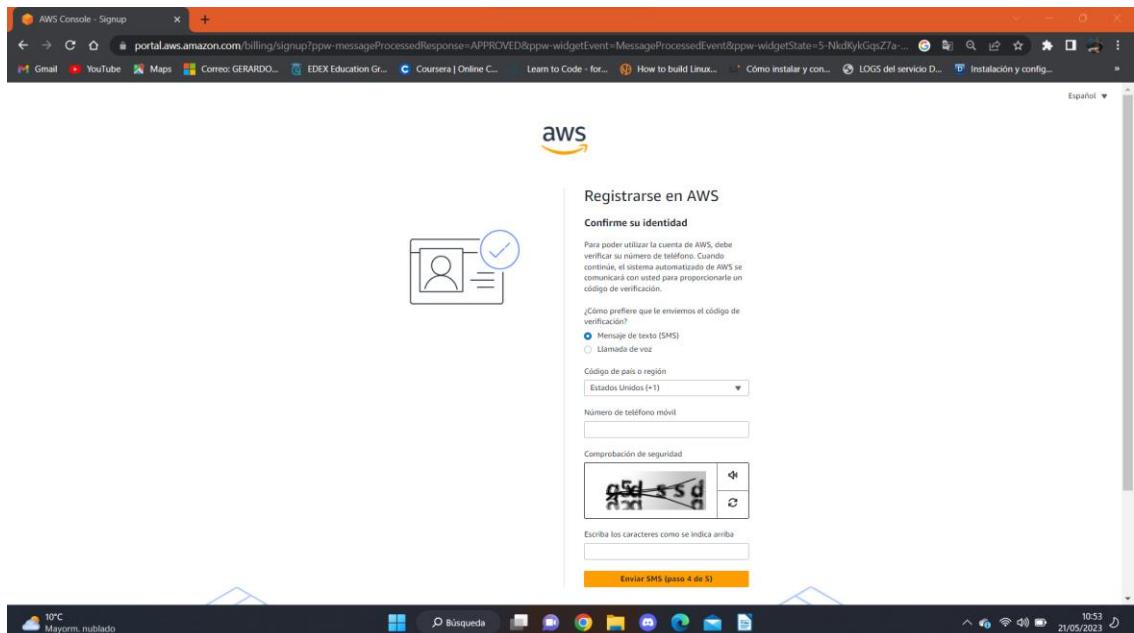
## Dockerización de un servidor completo en Cloud.



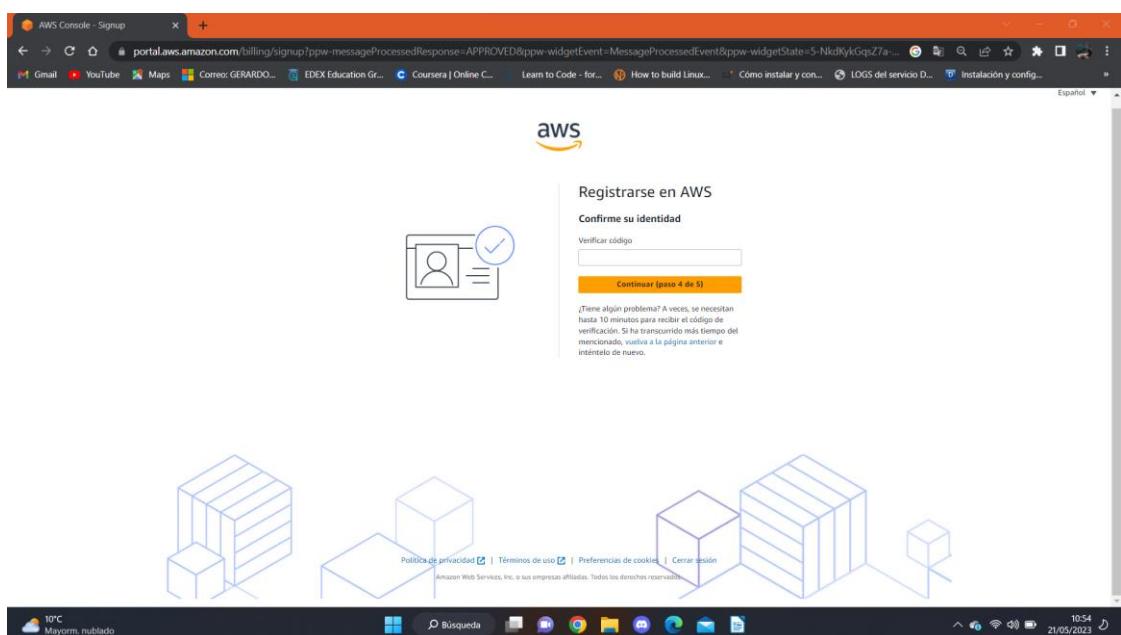
5. Seguidamente, se nos piden los datos de facturación. Es posible que se realice el cargo de 1\$ para confirmar nuestra identidad bancaria. Esta cantidad será reembolsada en un corto periodo de tiempo



6. En el siguiente paso del registro se nos pide verificar nuestro número de teléfono. Se puede llevar a cabo este proceso por mensaje de texto o por llamada. Marcamos la opción que prefiramos e introducimos de nuevo el teléfono. Resolvemos también la comprobación de seguridad.

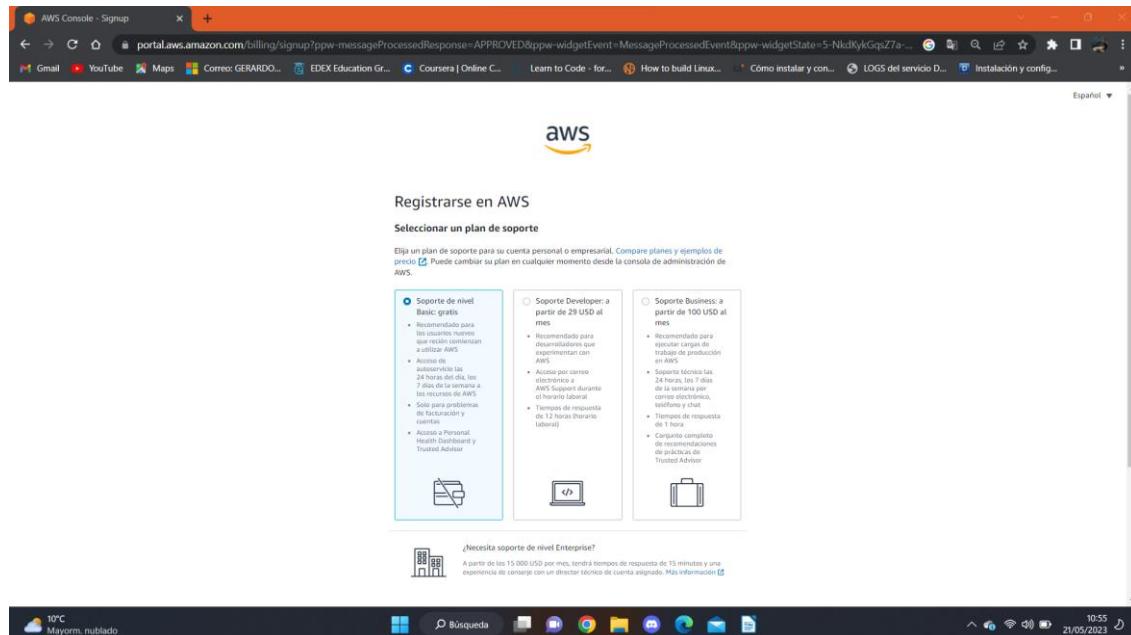


Una vez tengamos el código, lo introducimos.

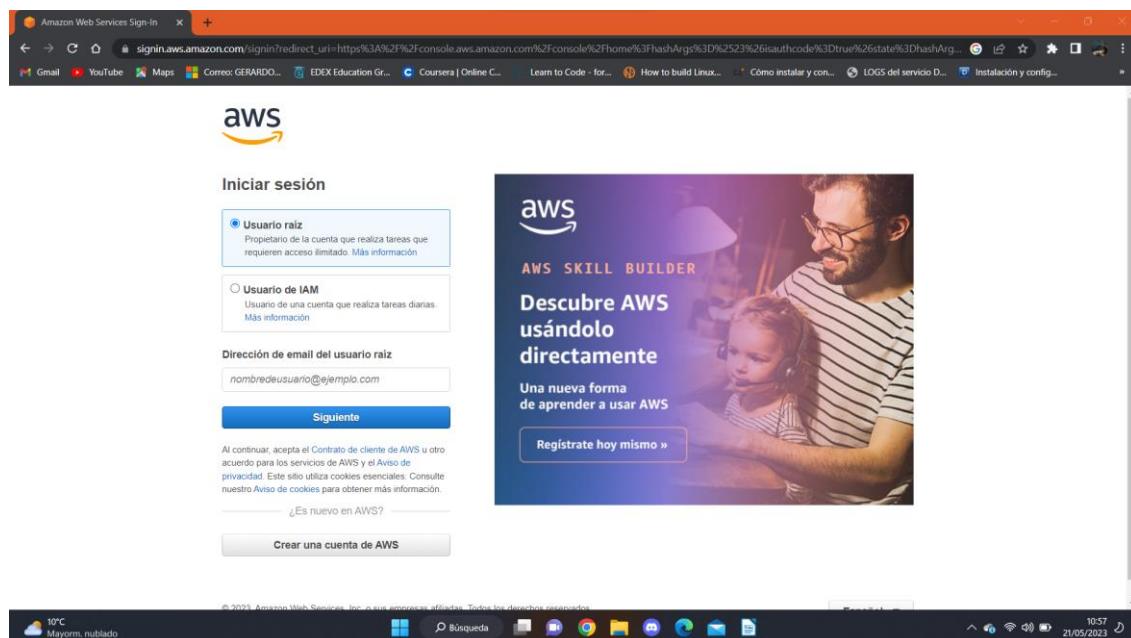


## Dockerización de un servidor completo en Cloud.

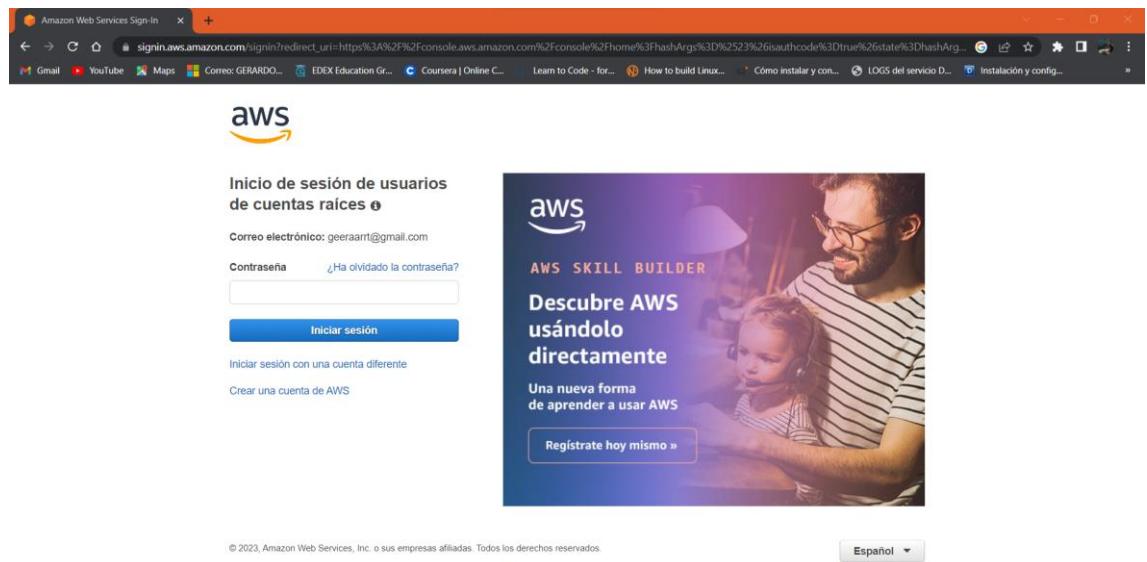
7. En el último paso, se nos pide elegir un plan de soporte. Sería importante realizar un estudio antes de seleccionar un plan, ya que debe ir adaptado a nuestro presupuesto y necesidades. En este caso, como utilizaremos la capa gratuita, seleccionamos el nivel ‘Basic gratis’.



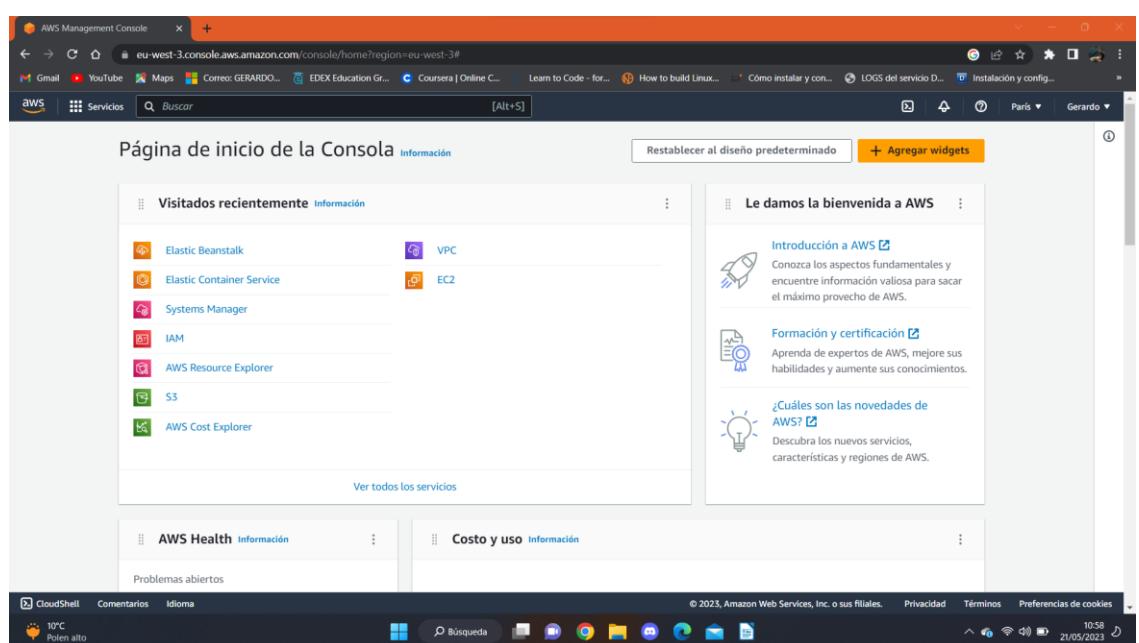
8. Con esto quedaría completado el registro. Para comenzar a utilizar recursos Cloud, pulsamos en ‘Ir a la consola de administración de AWS’. Se nos redirigirá a la página de inicio de sesión, en la que utilizaremos los datos que acabamos de registrar. Seleccionamos ‘Usuario raíz’ e introducimos el correo electrónico que registramos.



En la siguiente pantalla introducimos nuestra contraseña.



9. Se nos presentará la página de inicio de la Consola de AWS. Ya podremos hacer uso de los servicios incluidos en la capa gratuita de AWS.



Pulsa [aquí](#) para volver arriba.

## 10.3 Anexo 3: Manual de creación de la infraestructura.

En este manual vamos a seguir el proceso de creación de la infraestructura Cloud de red que permite la ejecución de nuestro proyecto.

Contamos con una red virtual (VPC) que tiene dos subredes: una pública, con acceso a Internet, y una privada, aislada del exterior. En la subred privada ubicamos el servidor, otorgándole así una mayor seguridad. En la subred pública ubicamos el bastión, que hará de mediador entre el exterior y el servidor. Por lo tanto, el bastión tiene acceso a Internet. Para este acceso a Internet es necesaria la creación de una Gateway de Internet. El bastión tiene asignada una IP elástica (fija). Las dos subredes se comunican mediante las rutas que se definen en las tablas de enrutamiento. El servidor, a pesar de estar en la subred privada sin Internet, recibe conexión a través del bastión, que actúa como servidor NAT. Para ello, se necesita la asignación de una Gateway NAT.

Vamos a comenzar con la creación de cada parte.

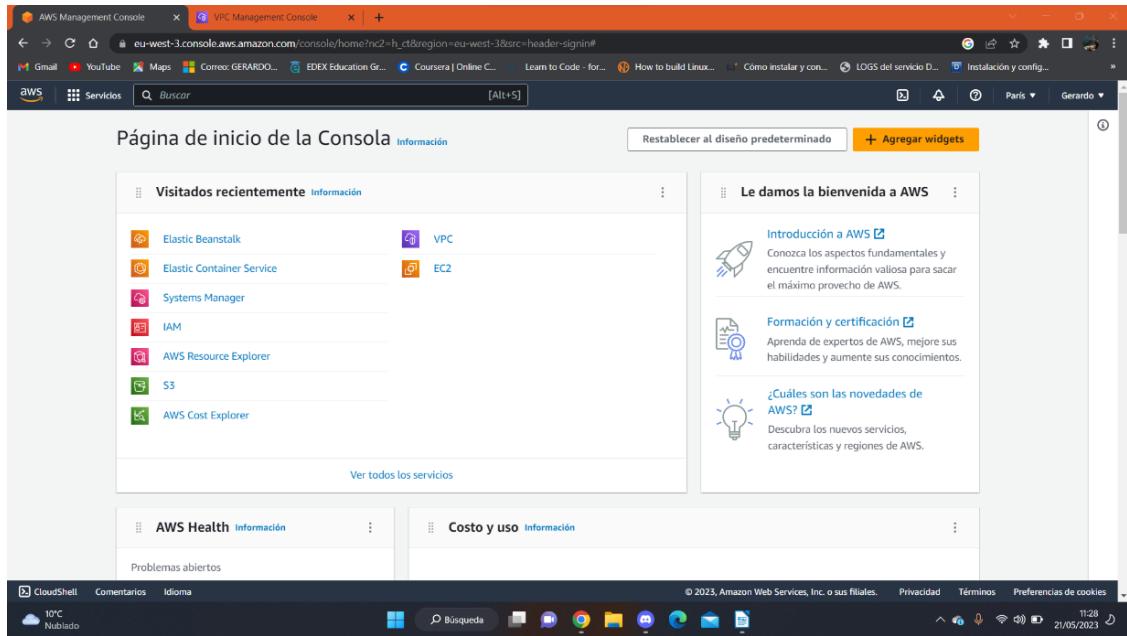
### VPC.

Una VPC (Virtual Private Cloud o Nube virtual privada), es, en esencia, una red virtual que permite al usuario crear y administrar su propia red en la nube de AWS.

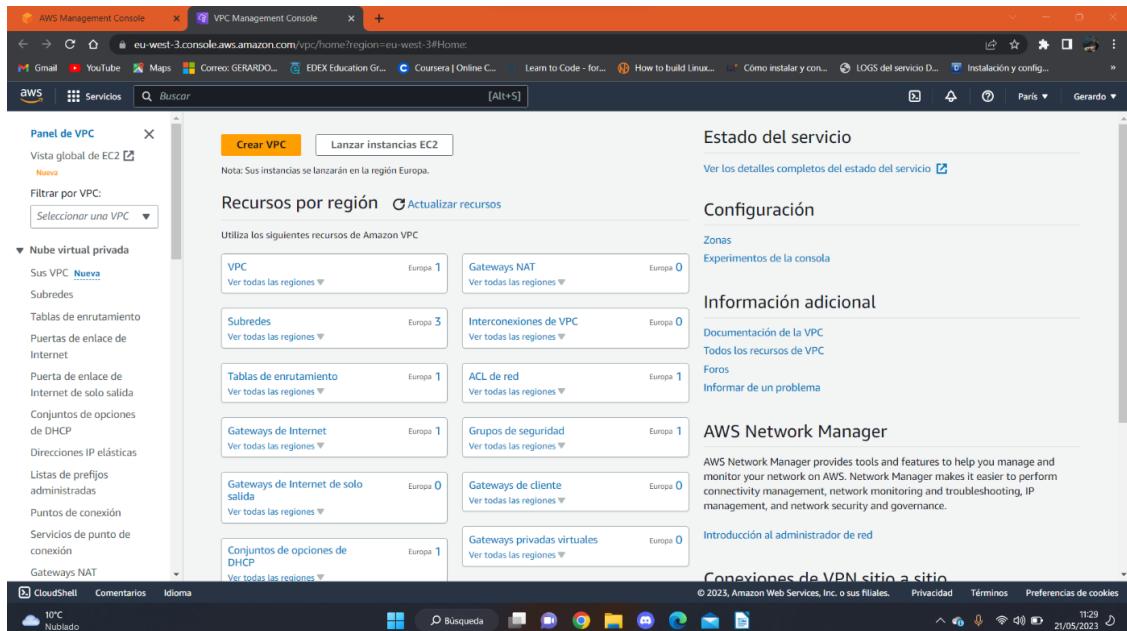
En esta red se pueden definir subredes, configurar tablas de enrutamiento y controlar la seguridad de los recursos de la nube. La amplia personalización de la VPC permite diseñar una infraestructura en la nube de acuerdo a las necesidades del usuario. Por defecto, la cuenta de AWS ya viene con una VPC creada para facilitar la creación de instancias. Nosotros vamos a crear una personalizada. Para evitar confusiones, cambiamos el nombre de la VPC por defecto a 'VPC\_por\_defecto'.

Para crear una VPC seguimos los siguientes pasos:

1. Desde la página de inicio de la consola de AWS, hacemos click en 'VPC'.

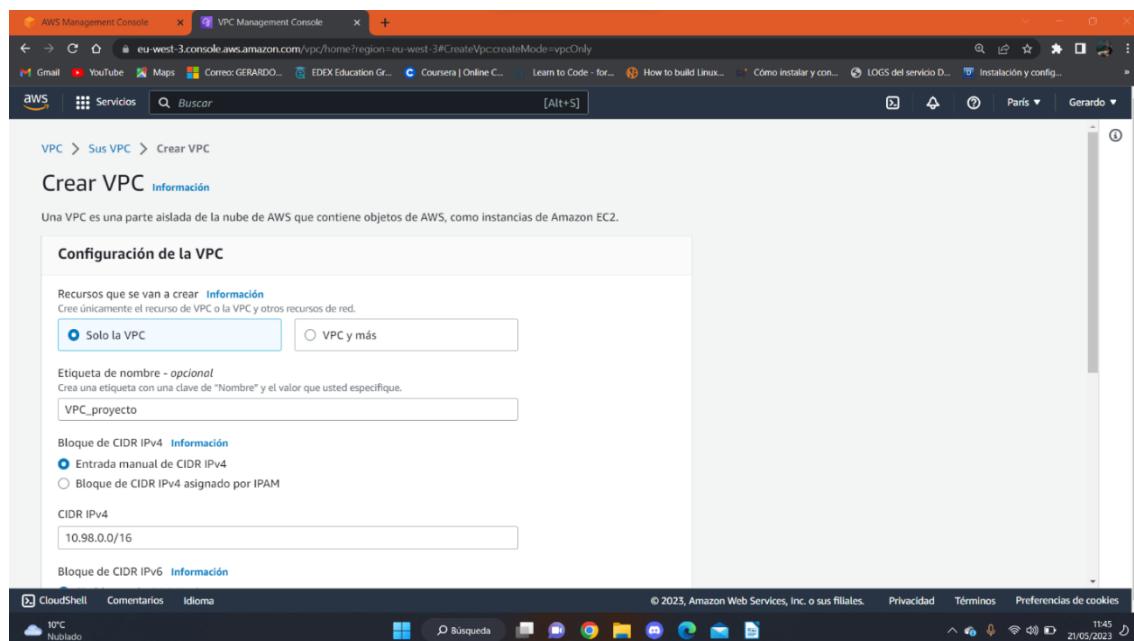


2. En el panel 'VPC', le damos a 'Crear VPC'. En este panel, además de iniciar el proceso de creación, encontramos el acceso a los submenús de los distintos recursos de VPC, como tablas de enrutamiento, subredes, puertas de enlace, etc.



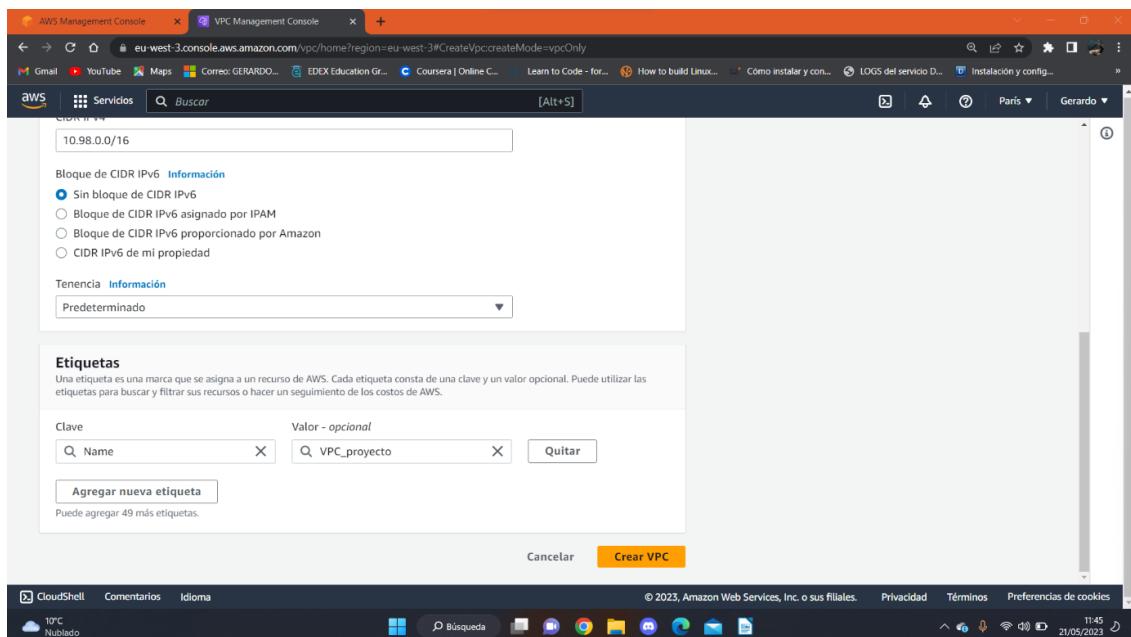
## Dockerización de un servidor completo en Cloud.

3. Seleccionamos ‘Solo la VPC’, ya que el resto de recursos los crearemos manualmente más tarde. Podemos añadir una etiqueta opcional que identifique la VPC. En este caso la hemos llamado “VPC\_proyecto”. A continuación, dejamos marcado la opción ‘Entrada manual de CIDR IPv4’, y en el cuadro siguiente introducimos la dirección y máscara de red que le queremos asignar a la VPC: “10.98.0.0/16”.

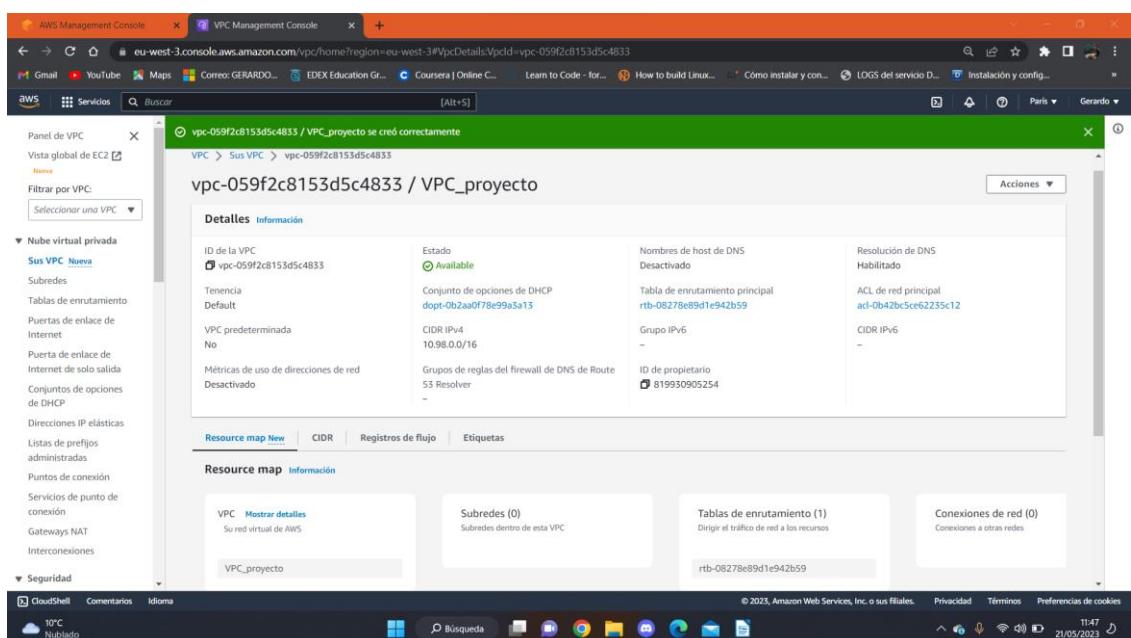


Como no vamos a utilizar IPv6, dejamos marcado ‘Sin bloque de CIDR IPv6’. Seguidamente, tenemos un apartado llamado Tenencia. La tenencia se refiere a la posibilidad de tener hardware físico completamente dedicado a el usuario. Esto incrementaría el precio del servicio, por lo tanto, como no nos es necesario, dejamos seleccionad el valor ‘Predeterminado’ y continuamos.

Por último, podemos agregar etiquetas a los recursos de AWS. Dichas etiquetas sirven para facilitar la búsqueda y filtrado de recursos o para hacer un seguimiento de los gastos. Como no vamos a tener una cantidad excesiva de recursos, dejamos la etiqueta por defecto 'Name' y damos a 'Crear VPC'.



4. Queda así creada nuestra VPC. Como podemos observar en la captura, se le asigna automáticamente una tabla de enrutamiento. Más tarde hablaremos de estas tablas



### Subredes.

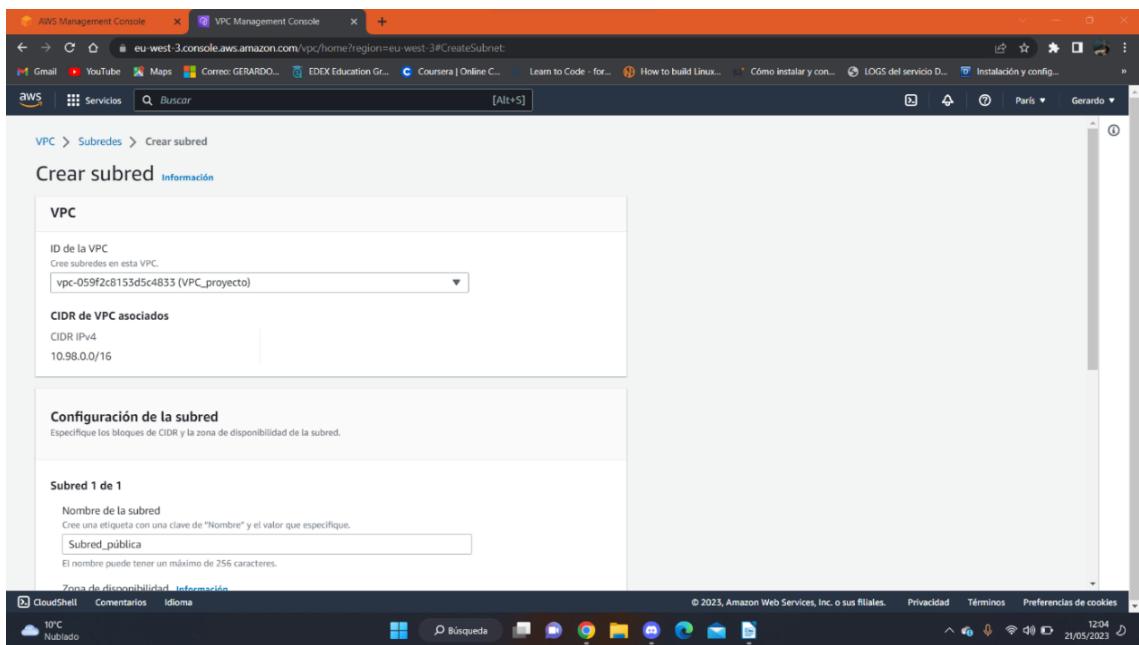
Una subred es una división de una red más grande. Permite organizar una red en redes más manejables, que cuentan con su propio intervalo de direcciones IP a las que conectar dispositivos. Suponen una mejora de rendimiento y de seguridad al limitar la cantidad de dispositivos que se alojan en la red. Al igual que las redes, se identifican con una dirección IP de subred y una máscara de red.

Por defecto, AWS trae definidas tres subredes para la VPC por defecto. Para evitar confusiones, se ha incluido en su nombre la expresión ‘por defecto’. Vamos a crear nuestras subredes:

1. Hacemos click en ‘Crear subred’:

Name	ID de subred	Estado	VPC	CIDR IPv4	CIDR IPv6	Dirección
Por_defecto1	subnet-0c9a5900a751d226	Available	vpc-0147845c99a09760c	172.31.16.0/20	-	4091
Por_defecto2	subnet-0914db66eab7e502e	Available	vpc-0147845c99a09760c	172.31.32.0/20	-	4091
Por_defecto3	subnet-0fc927976c41b102e	Available	vpc-0147845c99a09760c	172.31.0.0/20	-	4091

2. Seleccionamos la ID de la VPC a la que va a pertenecer la subred. Elegimos ‘VPC\_proyecto’. A continuación, introducimos un nombre para la subred. Esta subred, va a ser la que tenga salida a Internet, por lo tanto, la vamos a identificar como “Subred\_pública”.

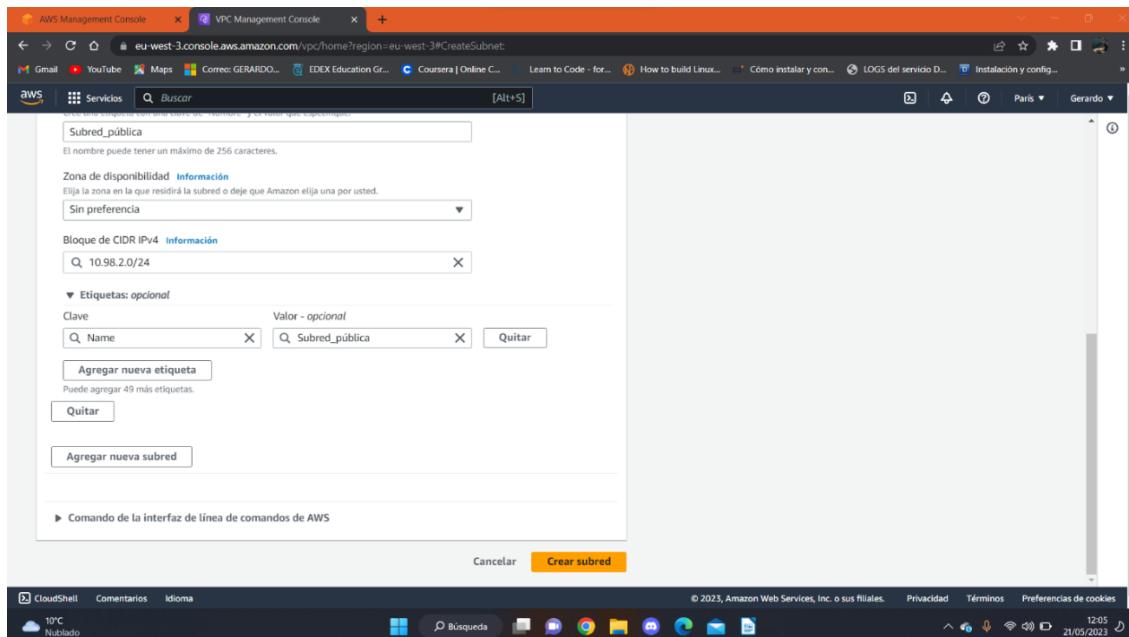


3. Seguidamente, seleccionamos la zona de disponibilidad. Como estamos trabajando en la de París, se nos ofrecen distintas zonas dentro de la propia zona de París. Como nos da igual, dejamos marcado ‘Sin preferencia’ para que AWS elija por nosotros.

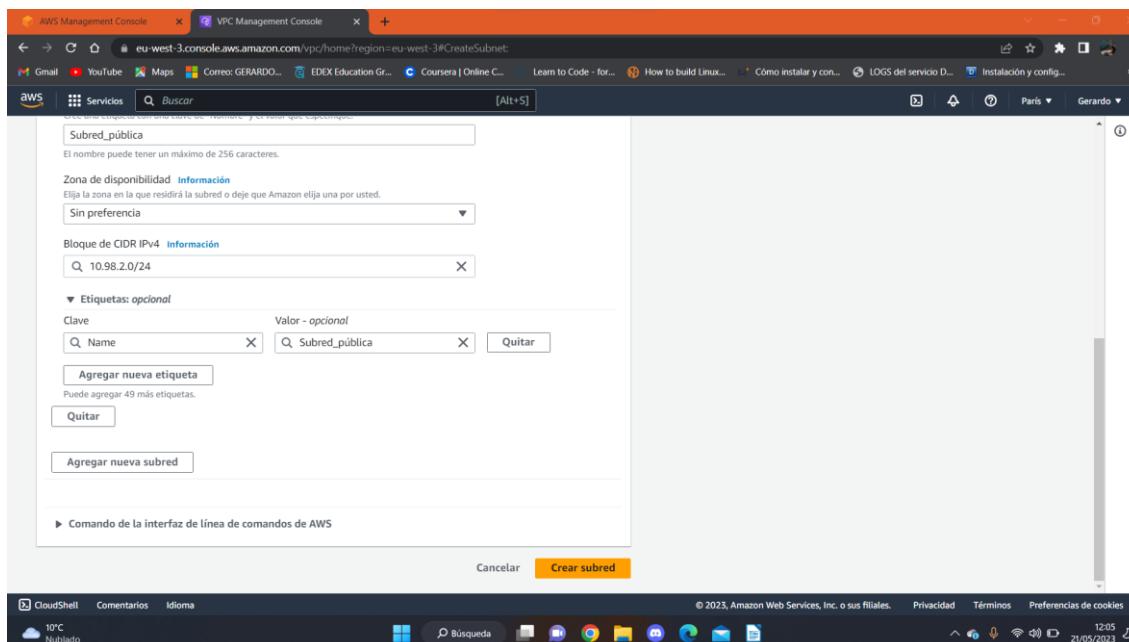
En ‘Bloque de CIDR IPv4’, introducimos la dirección de la subred y su máscara de red. Como no necesitamos un gran número de hosts posibles, nos hemos decantado por una máscara de red ‘/24’. La dirección, por lo tanto, será ‘10.98.2.0/24’.

## Dockerización de un servidor completo en Cloud.

Como en cada creación de recursos, se nos ofrece la posibilidad de añadir etiquetas. Dejamos la etiqueta ‘Name’ por defecto y finalmente hacemos click en ‘Crear subred’.



4. Ahora que conocemos el proceso de creación de subredes, creamos otra que nos servirá de subred privada sin salida a Internet. Siguiendo el mismo proceso punto por punto, sólo cambiamos el nombre de ‘Subred\_pública’ a “Subred\_privada” y en el ‘Bloque de CIDR IPv4’ introducimos ‘10.98.1.0/24’. Agregamos la subred y ya tenemos creadas las dos subredes necesarias para nuestra infraestructura



## Internet Gateway.

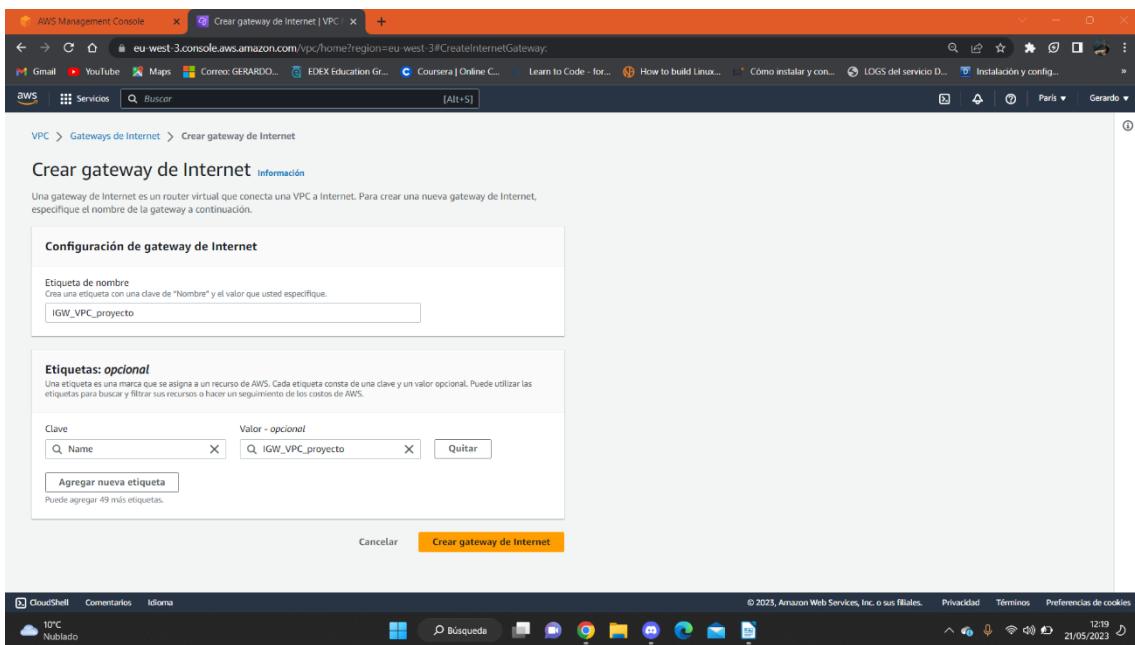
Una Puerta de enlace de Internet o Internet Gateway es un componente que permite la conectividad bidireccional entre una VPC e Internet. Funciona como punto de entrada y salida del tráfico de red. Por defecto, una VPC recién creada no tiene acceso a Internet. Para hacer posible dicha conexión, se le debe asociar una Internet Gateway. Esto no permite el acceso inmediato a Internet, si no que habría que definir las pertinentes rutas en las tablas de enrutamiento de las subredes. Esto se configurará más adelante.

1. Para crear la Internet Gateway, nos colocamos en el panel de VPC y hacemos click en ‘Gateways de Internet’. Como en los recursos anteriores, ya existe una puerta de enlace de este tipo para la VPC por defecto. Para evitar confusiones, le cambiamos el nombre a ‘IG\_por\_defecto’. Para crear una nueva, hacemos click en “Crear Gateway de Internet”.

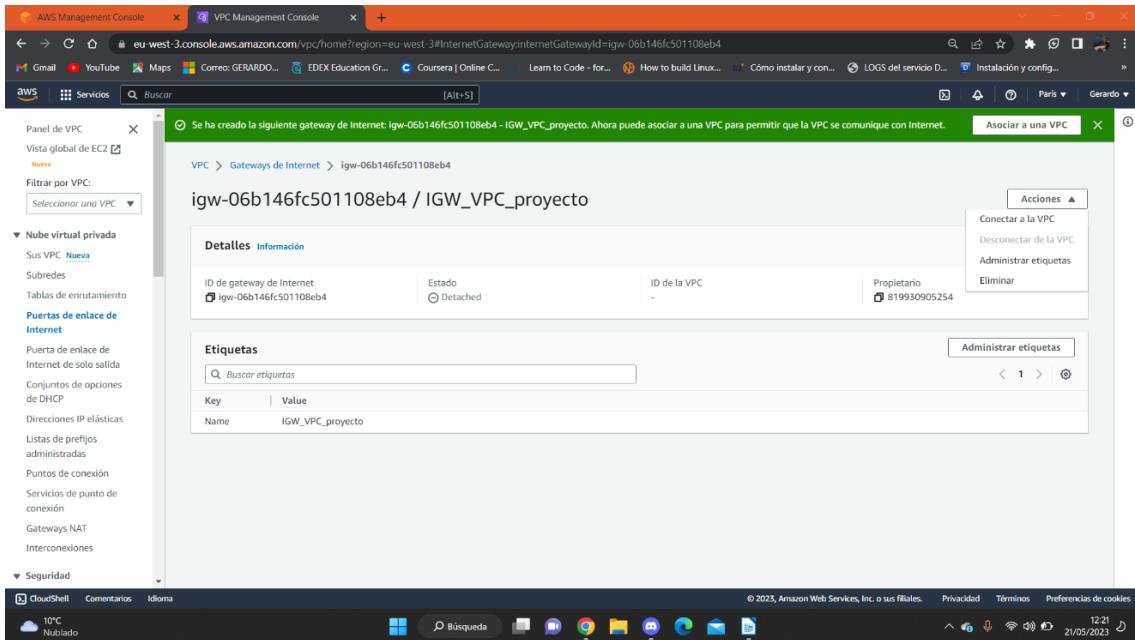
Name	ID de gateway de Internet	Estado	ID de la VPC	Propietario
IGW_por_defecto	igw-06e0819d59d909d15	Attached	vpc-0147845c99a09760c   VPC_por_d...	81993090524

## Dockerización de un servidor completo en Cloud.

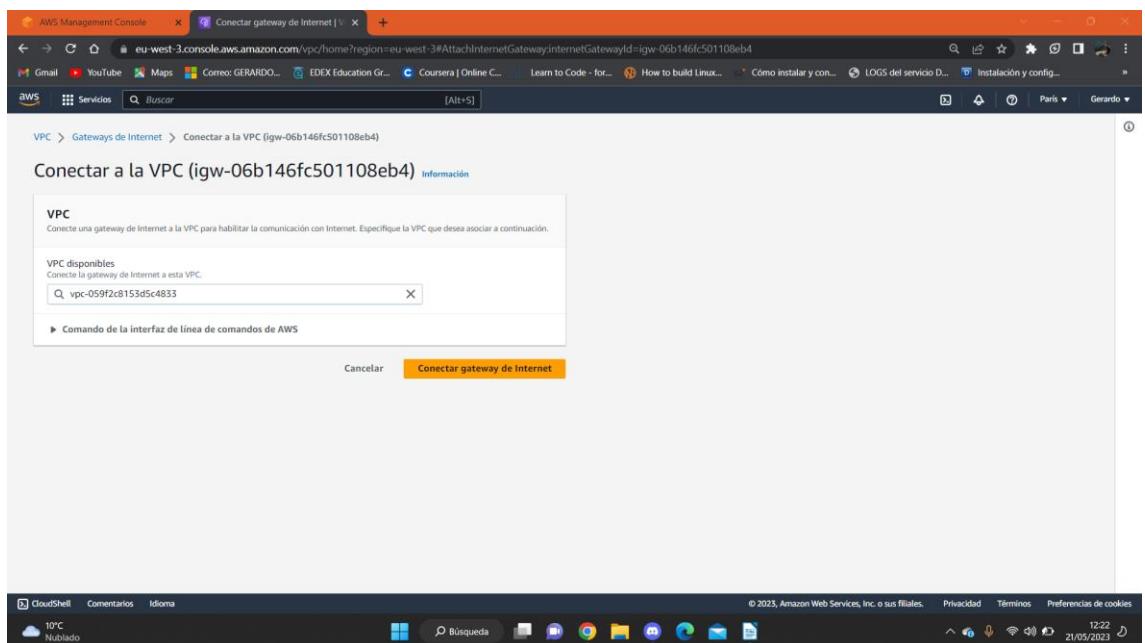
2. El proceso es muy sencillo, ya que sólo hay que asociarle un nombre, que será ‘IGW\_VPC\_proyecto’. Sin más, le damos a ‘Crear gateway de Internet’.



3. Seguidamente, hay que asociar esta Gateway recién creada a la VPC. Para ello, en la ventana a la que se nos conduce tras crear la Gateway, hacemos click en ‘Acciones’ y luego en ‘Conectar a la VPC’.



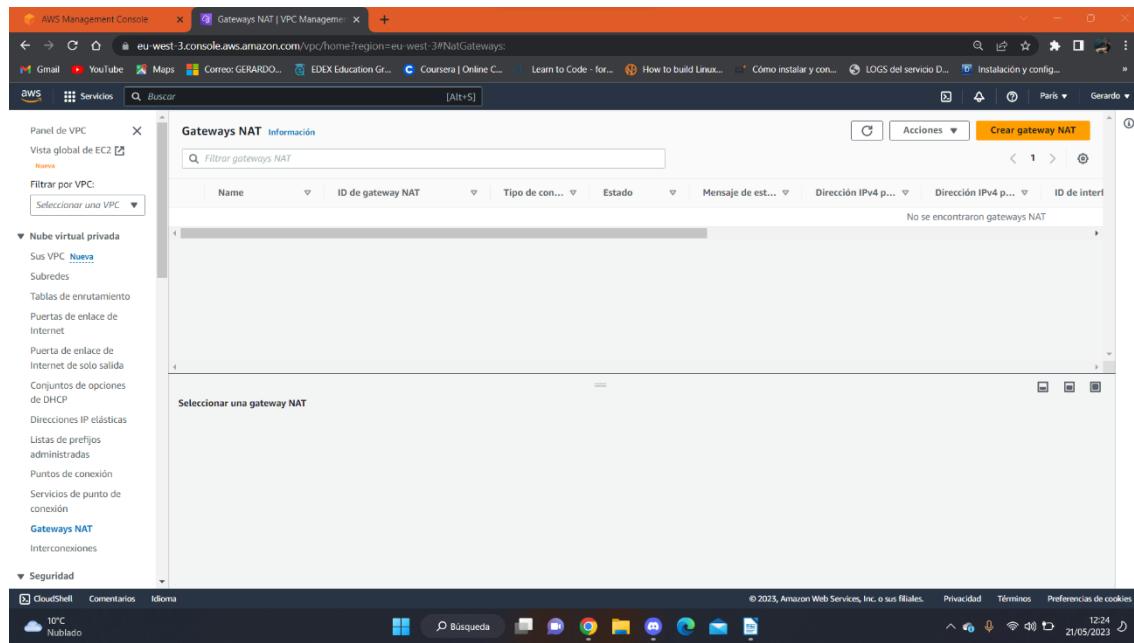
4. Ahora, simplemente seleccionamos nuestra VPC del proyecto y le damos a ‘Conectar gateway de Internet’. Queda así conectada la Gateway a la VPC.



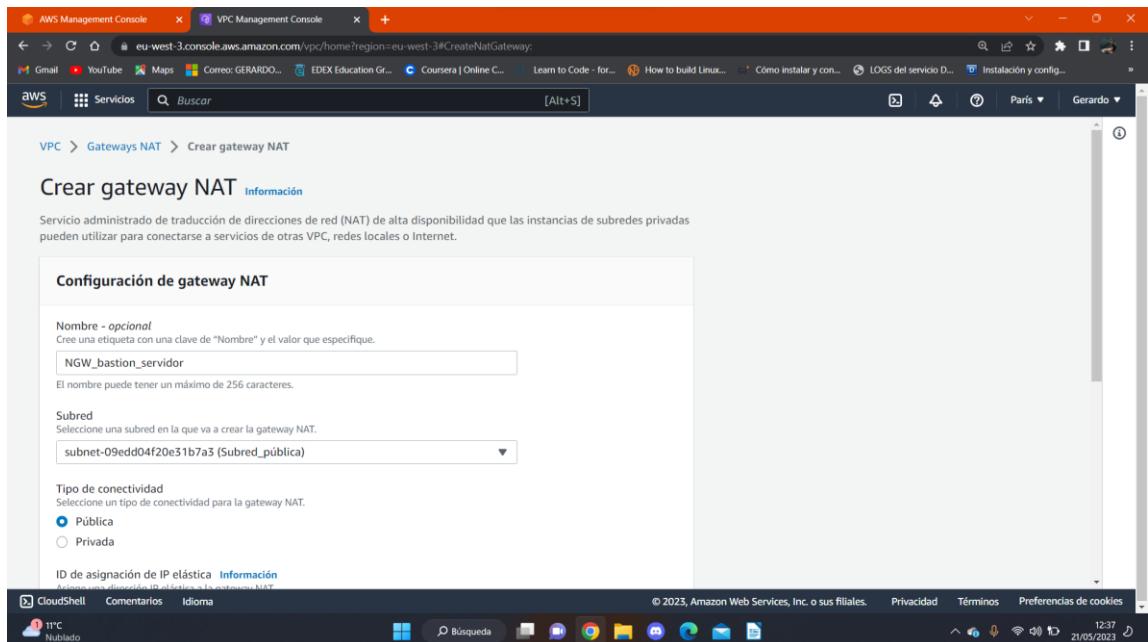
### Gateway NAT.

Una Gateway NAT es un componente que permite a un recurso de una VPC conectarse a Internet de manera segura. Esto nos hace falta para que nuestro servidor tenga acceso a Internet a través del bastión. Como la configuración de redes en las instancias se hace exclusivamente en AWS y no en sus archivos de configuración, necesitamos la Gateway NAT para habilitar el tráfico de red entre el bastión y el servidor.

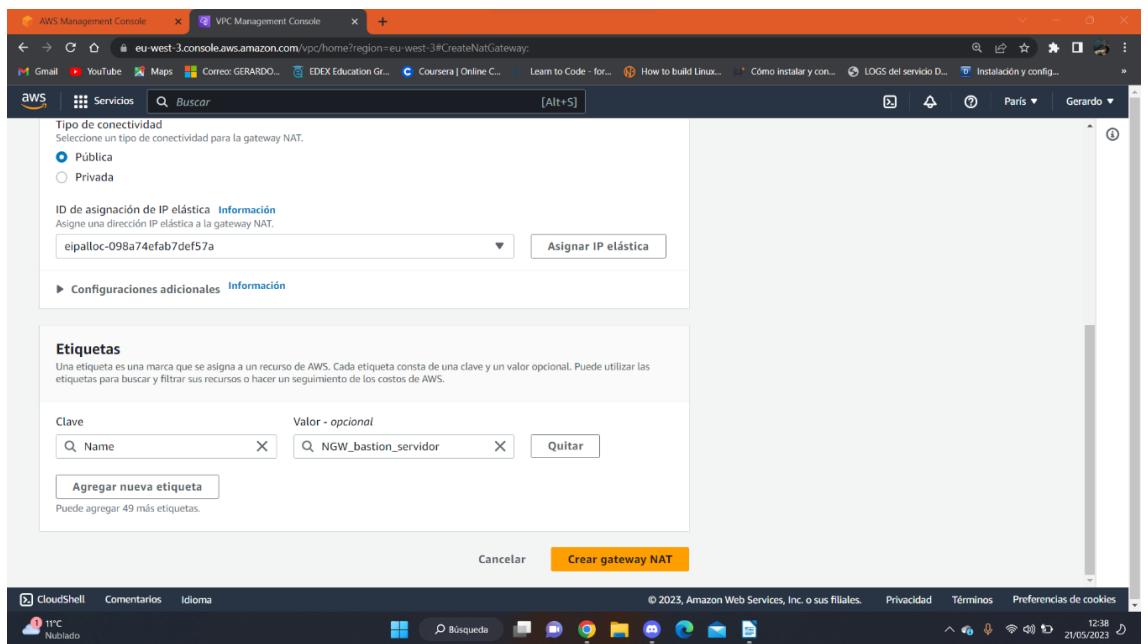
1. Para crear una Gateway Nat, nos colocamos en el panel general de VPC y hacemos click en ‘Gateways NAT’. Una vez en el panel de ‘Gateways NAT’, le damos a ‘Crear Gateway NAT’.



- Aquí, deberemos asignarle un nombre, que en este caso será ‘NGW\_bastion\_servidor’ y luego seleccionamos la subred a la que se asociará la Gateway NAT. En nuestro caso seleccionamos la subred pública. En ‘Tipo de conectividad’ dejamos marcada la opción por defecto ‘Pública’.



Ahora, tenemos que introducir un nuevo concepto de AWS. Se trata de las IPs elásticas. No es más que el nombre que Amazon le da a una IP fija. La Gateway NAT precisa de una de estas direcciones elásticas, ya que si cambiara cada vez que apagamos la infraestructura la conexión se echaría a perder. Por lo tanto, le damos a ‘Asignar IP elástica’ para crear una y asociarla a la Gateway NAT. Este paso es automático y no requiere de mayor configuración. Por último, hacemos click en ‘Crear gateway NAT’.



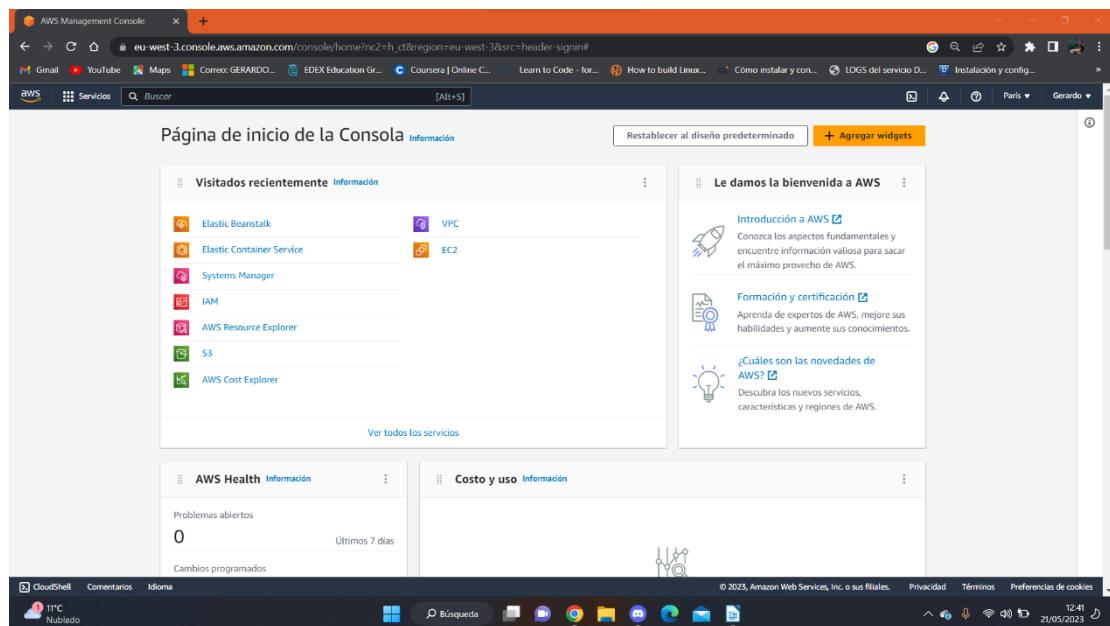
### Instancias.

Antes de hablar de las instancias, debemos comentar algunas cosas sobre temas de almacenamiento. La capa gratuita de AWS ofrece 30GB de almacenamiento, que pueden ser distribuidos sin coste entre los distintos servicios. De esta forma, el reparto del almacenamiento quedará distribuida de la siguiente forma:

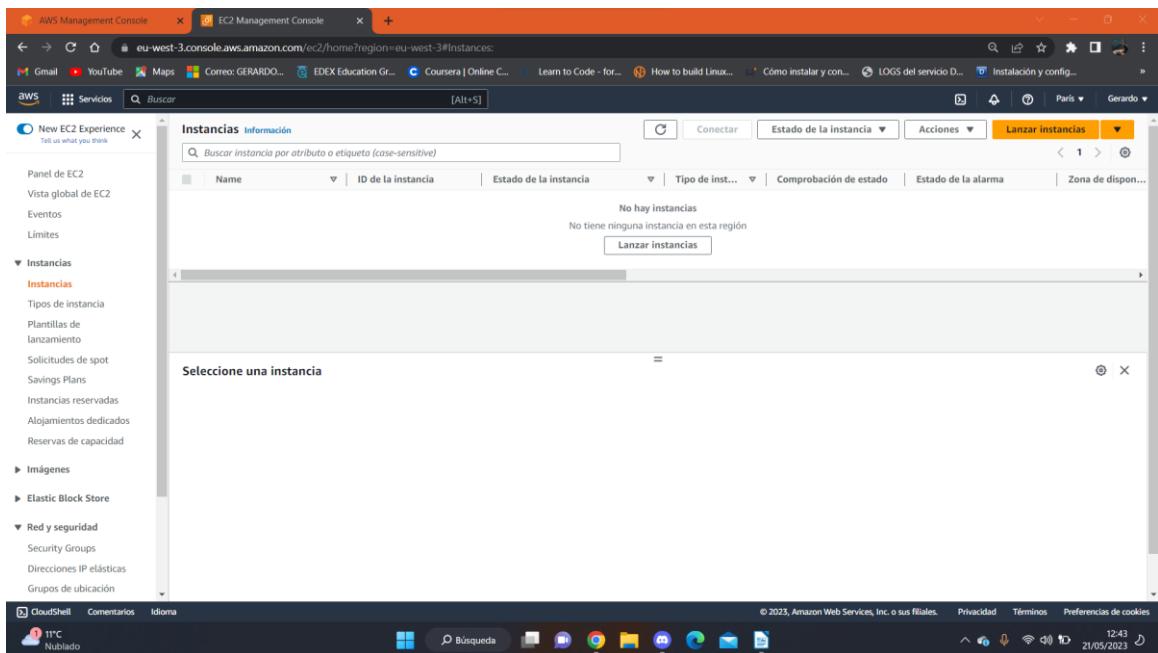
- 19GB para la instancia del servidor.
- 8GB para la instancia del bastión.
- 1GB para el bucket s3. De este servicio hablaremos más adelante.
- 2GB de reserva.

Ahora sí, vamos a pasar a crear las instancias EC2 con las que vamos a trabajar. Una instancia EC2 es el análogo en AWS de la clásica máquina virtual que conocemos y con las que hemos trabajado con VirtualBox.

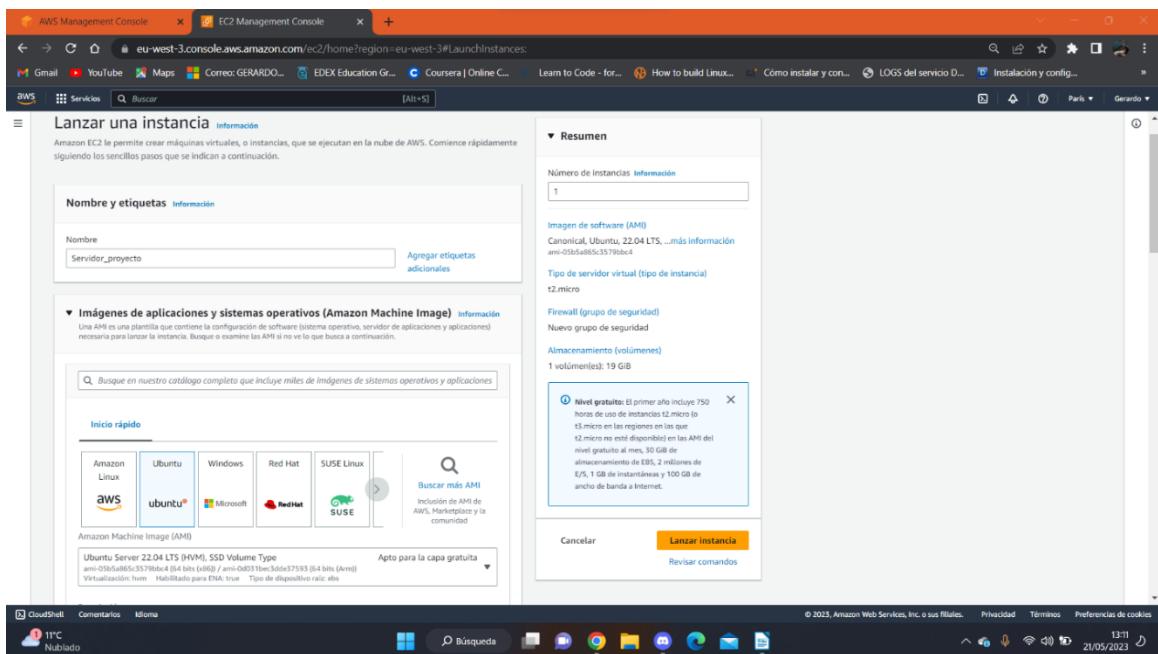
1. Para crear una instancia, nos dirigimos al panel de la consola de AWS y seleccionamos 'EC2'.



2. Una vez en el panel ‘EC2’, hacemos click en ‘Instancias’ y nos llevará a la pantalla de administración de Instancias. Aquí, damos a ‘Lanzar instancias’ para crear nuestra primera instancia. Empezaremos por el servidor.

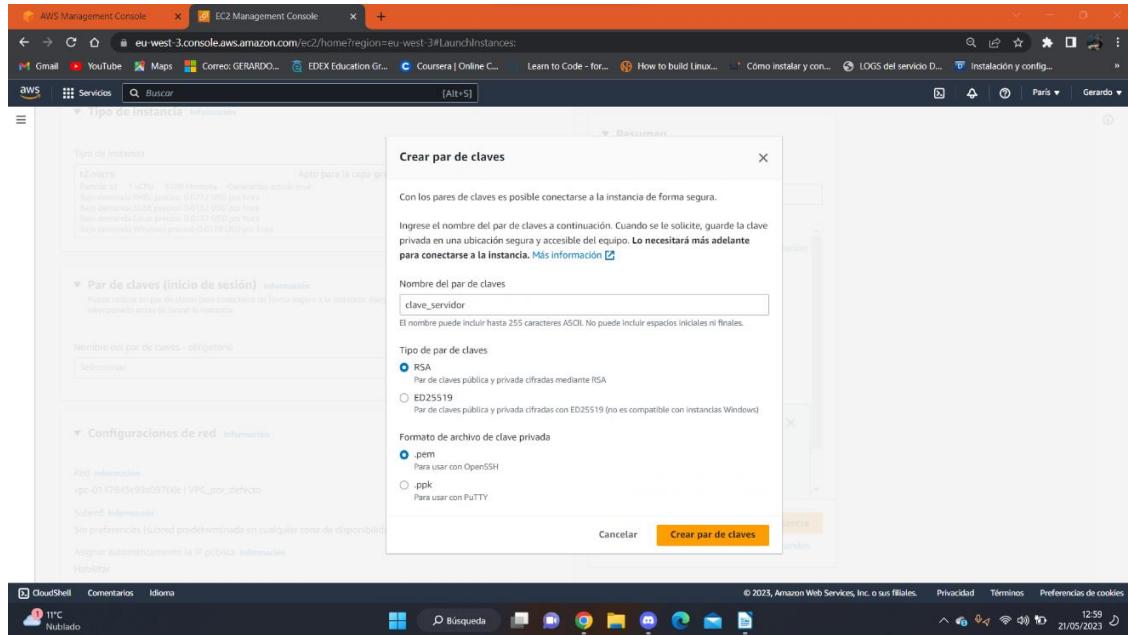


3. Comenzamos asignándole un nombre, que será ‘Servidor\_proyecto’. Seguidamente, seleccionamos la imagen del sistema operativo que vamos a utilizar. En nuestro caso, escogemos ‘Ubuntu’ y dejamos la versión por defecto ‘Ubuntu Server 22.04 LTS’ que viene incluida en la capa gratuita de AWS.

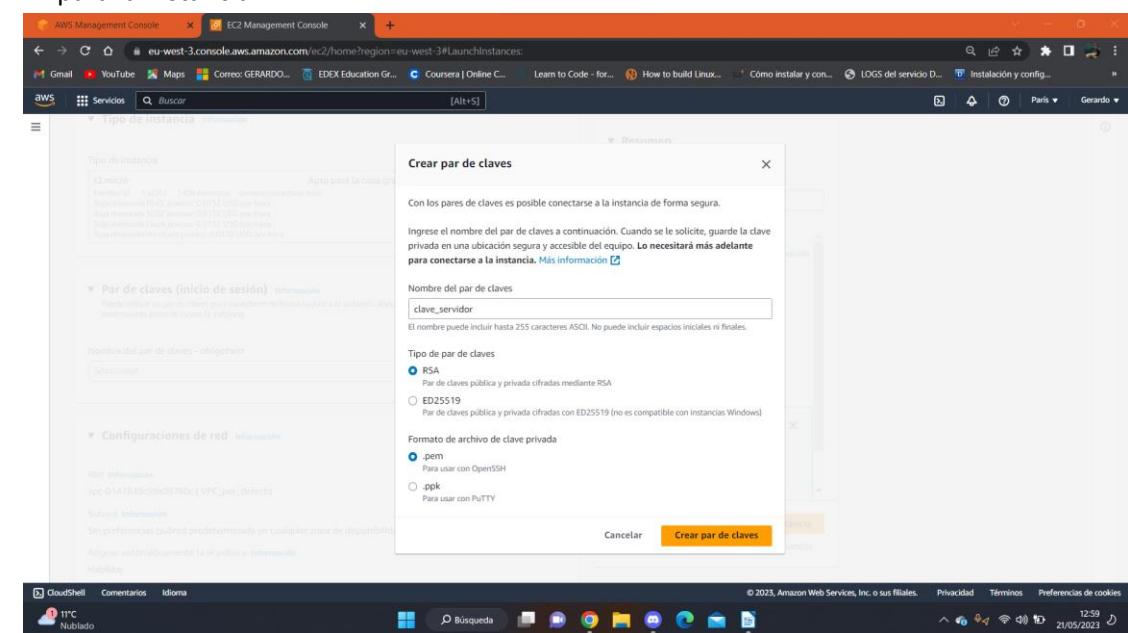


## Dockerización de un servidor completo en Cloud.

4. En ‘Arquitectura’, dejamos el valor por defecto ‘64 bits (x86)’. En tipo de instancia, dejamos seleccionado el valor ‘t2.micro’, ya que es el que está incluido en la capa gratuita. Este tipo de instancia cuenta con 1GB de RAM y un solo procesador, poco pero suficiente para nuestro humilde servidor. Necesitaremos un par de claves para el futuro uso de la conexión SSH. AWS nos facilita la creación de estas claves en un simple click



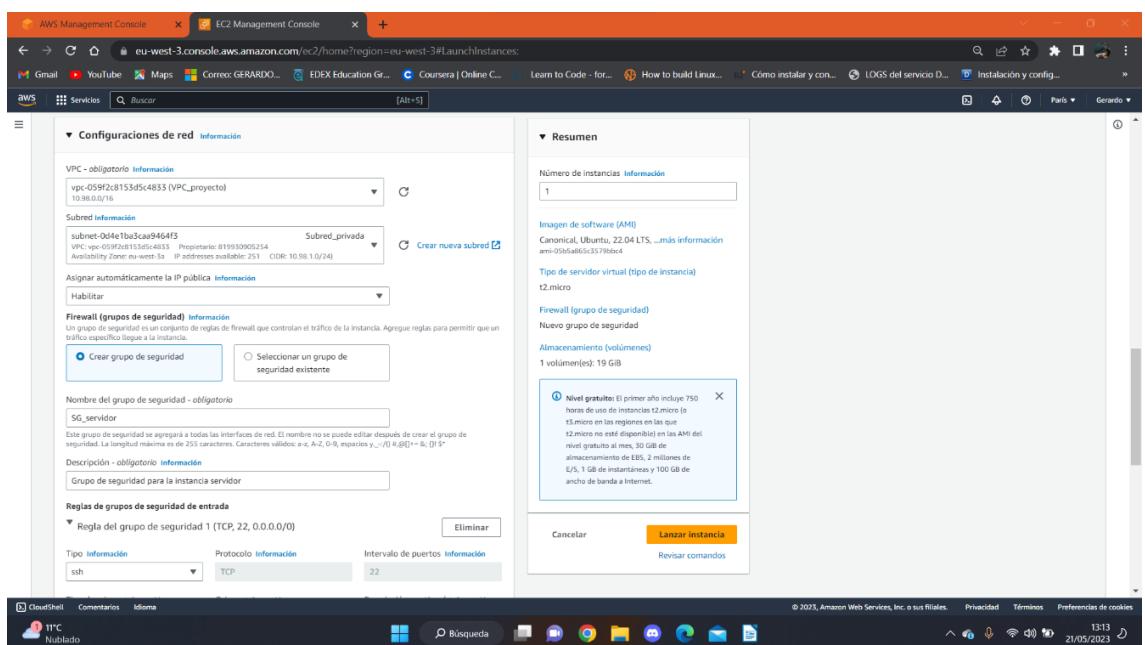
5. En el apartado ‘Par de claves (inicio de sesión)’, pulsamos en ‘Crear un nuevo par de claves’. Se nos abre una ventana en la que se nos pide nombrar el par de claves. El nombre seleccionado será “clave\_servidor”. En tipo de par de claves y en formato de archivo dejamos las opciones predeterminadas. Hacemos click ‘Crear par de claves’ y se nos descarga inmediatamente el fichero que contiene las claves. Deberemos guardarlo y tenerlo localizado para futuro uso. Automáticamente, el par de claves creado se selecciona para la instancia.



- A continuación, vamos con la configuración de red. Pulsamos ‘Editar’ y empezamos seleccionando la VPC del proyecto. Seguidamente, seleccionamos la subred adecuada. El servidor irá en la privada, así que la seleccionamos.

En ‘Asignar automáticamente la IP pública’, da igual el valor que seleccionemos, ya que el tráfico de red del servidor saldrá al exterior con la IP pública del bastión. De todas formas, seleccionamos ‘Habilitar’ para evitar posibles errores futuros.

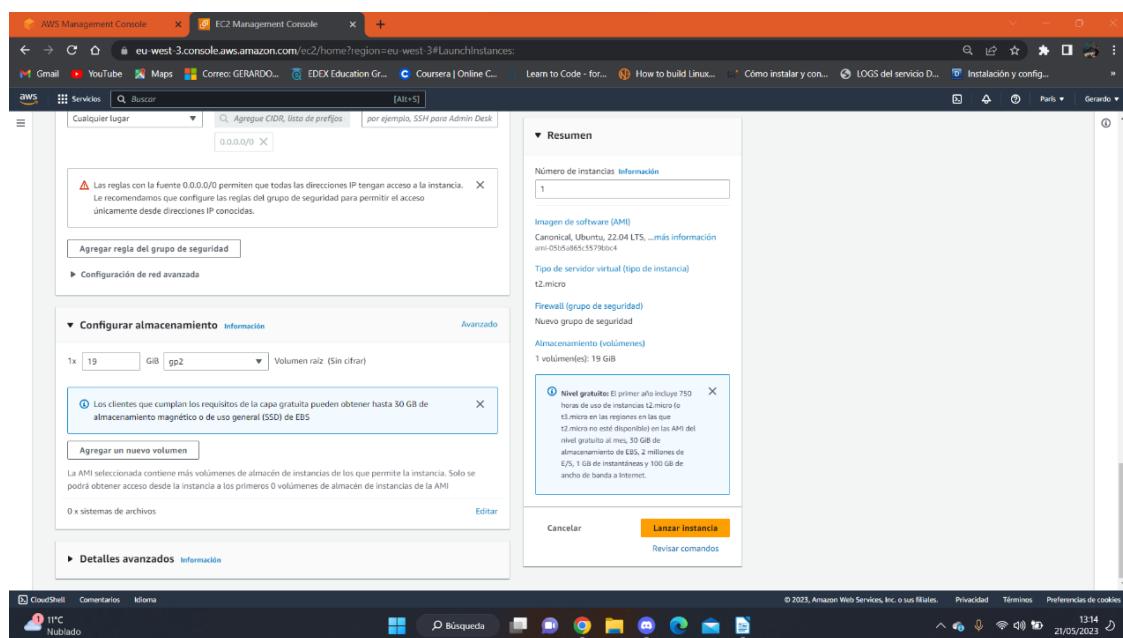
En ‘Firewall (grupos de seguridad)’, marcamos ‘Crear grupo de seguridad’. Este grupo de seguridad será el encargado de la configuración del tráfico por los distintos puertos de la instancia. Dicha configuración se realizará más adelante. De momento sólo le asignamos el nombre “SG\_servidor” y le añadimos una descripción obligatoria. Como ya dije antes, las reglas del grupo se añadirán más tarde, en este momento sólo dejamos las predeterminadas.



## Dockerización de un servidor completo en Cloud.

7. En configurar almacenamiento, vamos a seguir las directrices que comentamos en la introducción de la descripción de la infraestructura. Como dijimos, seleccionamos 19GB para el servidor. Dejamos el valor ‘gp2’ por defecto, ya que es el tipo de almacenamiento incluido en la capa gratuita.

Con esto, termina esta configuración inicial de la instancia. Procedemos a su lanzamiento haciendo click en ‘Lanzar instancia’. Con esto queda creada la instancia del servidor.



Ahora, vamos con la creación de la instancia del bastión. Será también un Ubuntu 22.04 con las mismas características. Como el proceso es el mismo, sólo vamos a comentar las diferencias.

Como le asignamos “Bastion\_proyecto”. Seguimos el mismo proceso para la creación de claves, en la que simplemente cambiamos el nombre del fichero de claves a “clave\_bastion”. En la configuración de red, la VPC será la misma, pero en subred elegimos la subred pública. Creamos un nuevo grupo de seguridad como lo hicimos antes y le asignamos el nombre “SG\_bastion”. En almacenamiento, el bastión no requiere tanto como el servidor. Por lo tanto, dejamos el valor por defecto de 8GB y lanzamos la instancia.

En la pantalla del menú de las instancias, éstas nos quedarían de la siguiente manera, ya funcionando

Recursos	Vista global de EC2	C	O
Actualmente, utiliza los siguientes recursos de Amazon EC2 en la región Europa (París):			
Instancias (en ejecución)	2	Auto Scaling Groups	0
Direcciones IP elásticas	1	Grupos de seguridad	4
Hosts dedicados	0	Grupos de ubicación	0
Pares de claves	2	Instancias	2
		Instantáneas	0
		Volumenes	2

**Lanzar la instancia**  
Para comenzar, lance una instancia de Amazon EC2, que es un servidor virtual en la nube.

**Estado del servicio**  
Región: Europa (París) Estado: Este servicio funciona con normalidad

**Zonas**  
Nombre de la zona ID de la zona  
eu-west-3a euw3-az1  
eu-west-3b euw3-az2  
eu-west-3c euw3-az3

**Eventos programados**  
Europa (París)  
No hay eventos programados

**Atributos de la cuenta**

- Plataformas compatibles
  - VPC
  - VPC predeterminada vpc-0147845c99a09760c
- Configuración
  - Cifrado de EBS
- Zonas
  - Consola de serie de EC2
  - Especificación de crédito predeterminada
  - Experimentos de la consola

**Información adicional**

- Guía de introducción
- Documentación
- Todos los recursos de EC2
- Foros
- Precios
- Póngase en contacto con nosotros

**Temas de ayuda**

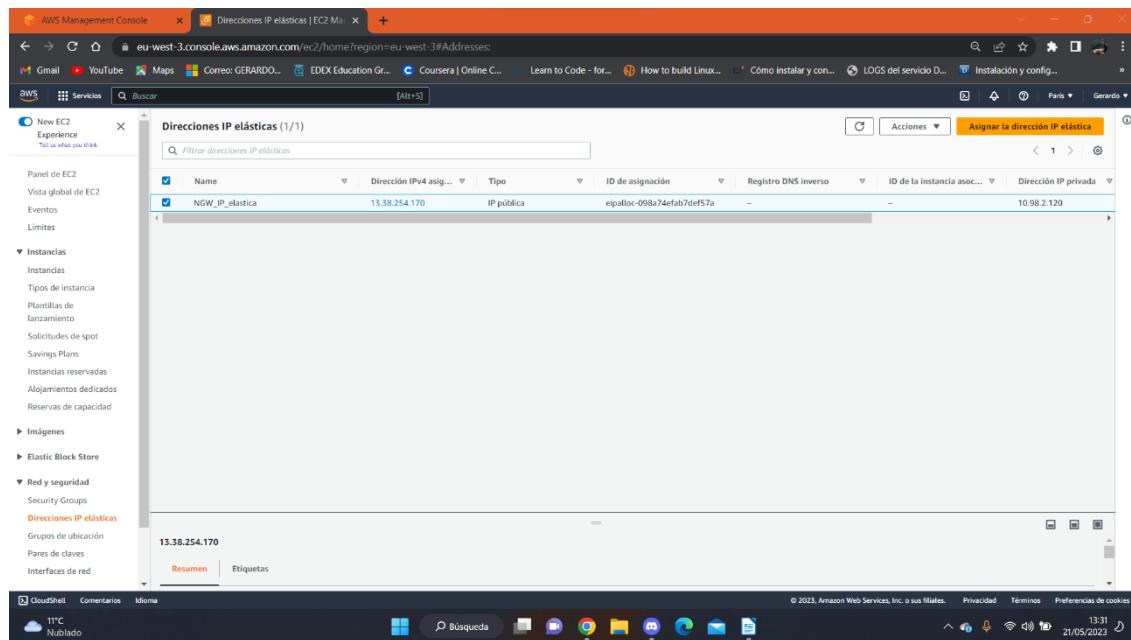
Otra diferencia entre las instancias es que el bastión requiere de una IP elástica (fija) para que cuando se apague la instancia, su IP pública no cambie, y así asegurarse de que la configuración en cuanto a la solicitud de servicios no tenga que cambiar. Dicho de otra forma, una IP elástica nos permite hacer las peticiones de servicios siempre a la misma IP. Para asignarla nos colocamos en el ‘Panel de EC2’ y hacemos click en ‘Direcciones IP elásticas’.

Instancias (2) Información									
<input type="text"/> Buscar instancia por atributo o etiqueta (case-sensitive)									
Name	ID de la instancia	Estado de la instancia	Tipo de inst...	Comprobación de estado	Estado de la alarma	Zona de dispon...	DNS de IPv4 público		
Servidor_proyecto	i-0769e1aba1baacf9	En ejecución	t2.micro	2/2 comprobaciones superadas	Sin alarmas	+ eu-west-3a	-		
Bastion_proyecto	i-04b63f075e9190964	En ejecución	t2.micro	2/2 comprobaciones superadas	Sin alarmas	+ eu-west-3a	-		

**Selección de una instancia**

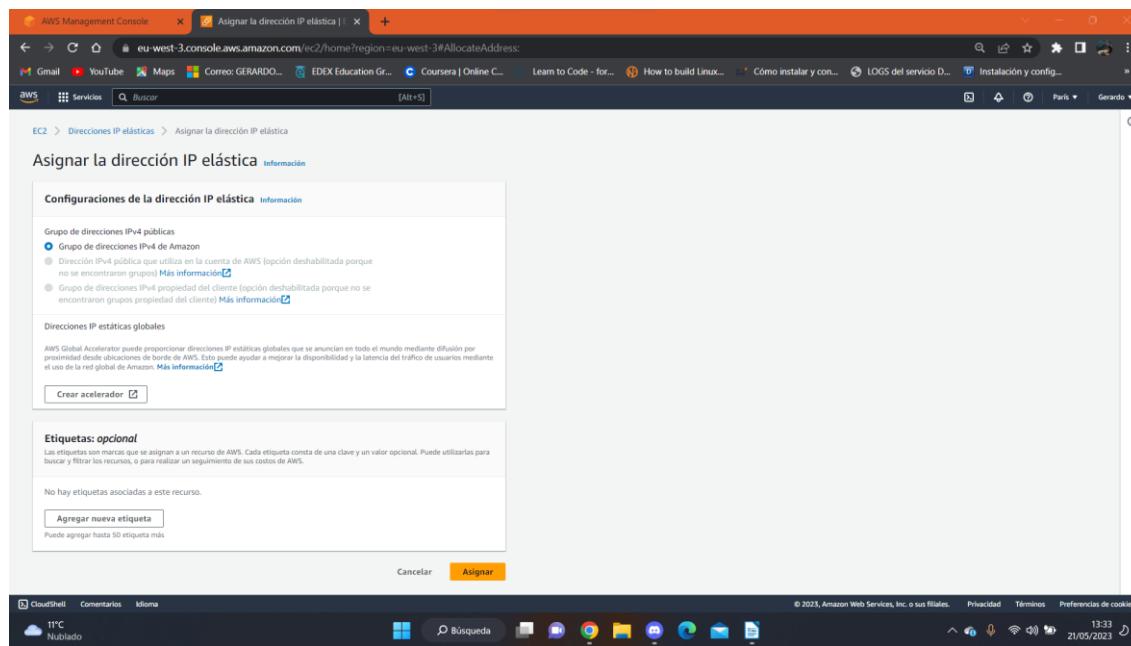
## Dockerización de un servidor completo en Cloud.

En la nueva ventana, seleccionamos ‘Asignar la dirección IP elástica’.



The screenshot shows the AWS Management Console interface for managing elastic IP addresses. On the left, there's a sidebar with various services like EC2, S3, and Lambda. The main area is titled 'Direcciones IP elásticas (1/1)'. It lists one item: 'NGW\_IP\_elastica' with IP address '13.38.254.170' and type 'IP pública'. At the top right of this list, there's a yellow button labeled 'Asignar la dirección IP elástica'. Below the list, there's a small window showing the IP address '13.38.254.170'.

En la pantalla que se nos presenta lo dejamos todo como está y damos a ‘Asignar’.



The screenshot shows the 'Allocate Address' configuration page. It has two main sections: 'Configuraciones de la dirección IP elástica' and 'Etiquetas: optional'. In the first section, 'Grupo de direcciones IPv4 públicas' is selected. In the second section, there's a note about AWS Global Accelerator and a 'Crear acelerador' button. At the bottom, there's a 'Cancelar' button and a prominent yellow 'Asignar' button.

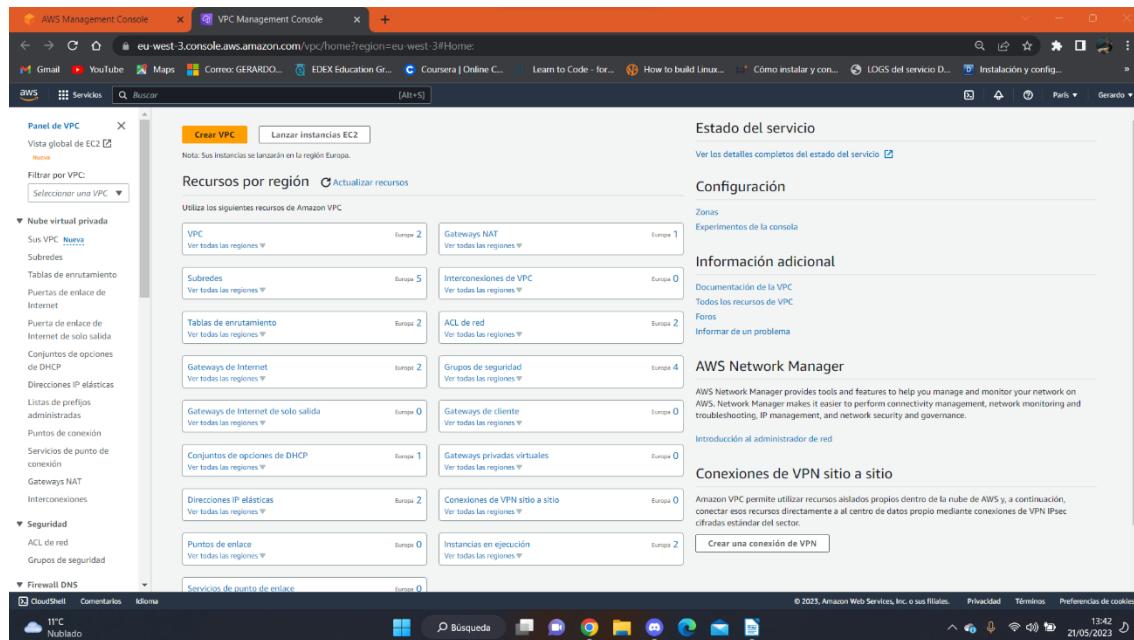
Le damos un nombre a la IP elástica para evitar confusiones (Bastion\_IP\_elastica) y a continuación, con la IP seleccionada, damos a ‘Acciones’ y a ‘Asociar la dirección IP elástica’.

En ‘Tipo de recurso’ dejamos seleccionado ‘Instancia’ y en ‘Instancia’ elegimos la instancia bastión. En ‘Dirección IP privada’ elegimos la IP privada del bastión y en ‘Nueva asociación’ no tocamos nada. Finalmente, damos a ‘Asociar’ y quedarán ya vinculadas la instancia y su IP elástica.

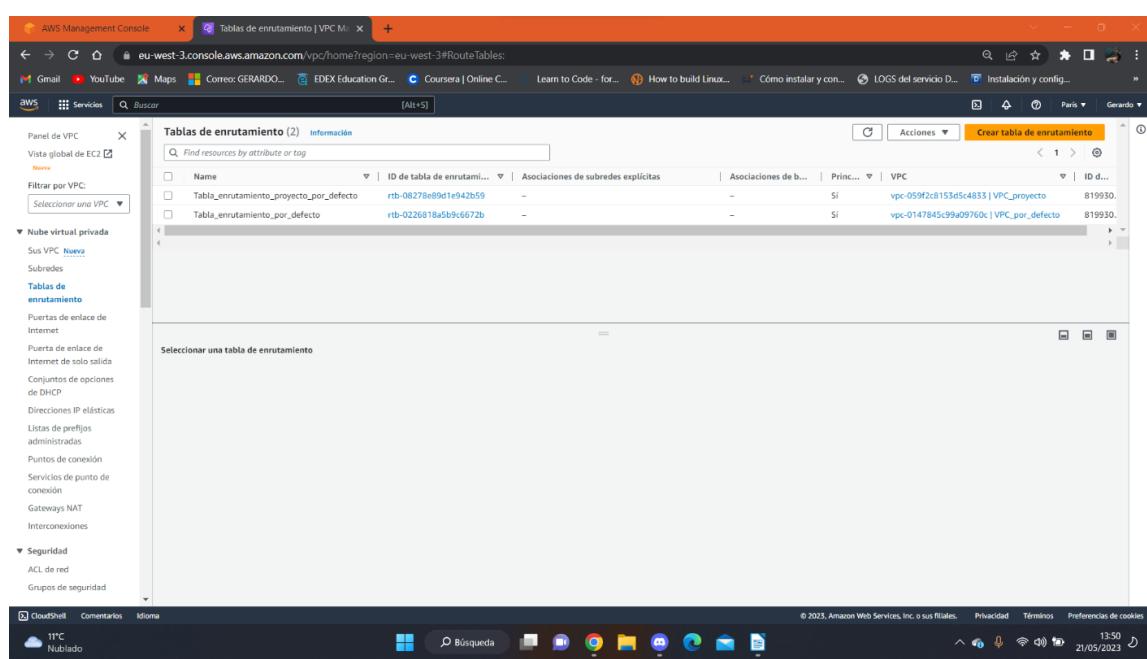
### Tablas de enrutamiento.

Las tablas de enrutamiento son una parte vital de la configuración de red de las VPC. Son un conjunto de reglas que determinan cómo se enruta el tráfico dentro de una red o entre redes. Se configuran a nivel de subred y se pueden asociar a una o varias subredes de una VPC. Se usan, junto a los grupos de seguridad (que se configurarán en el manual de cada servicio), para lograr que la red sea tan segura como el usuario determine.

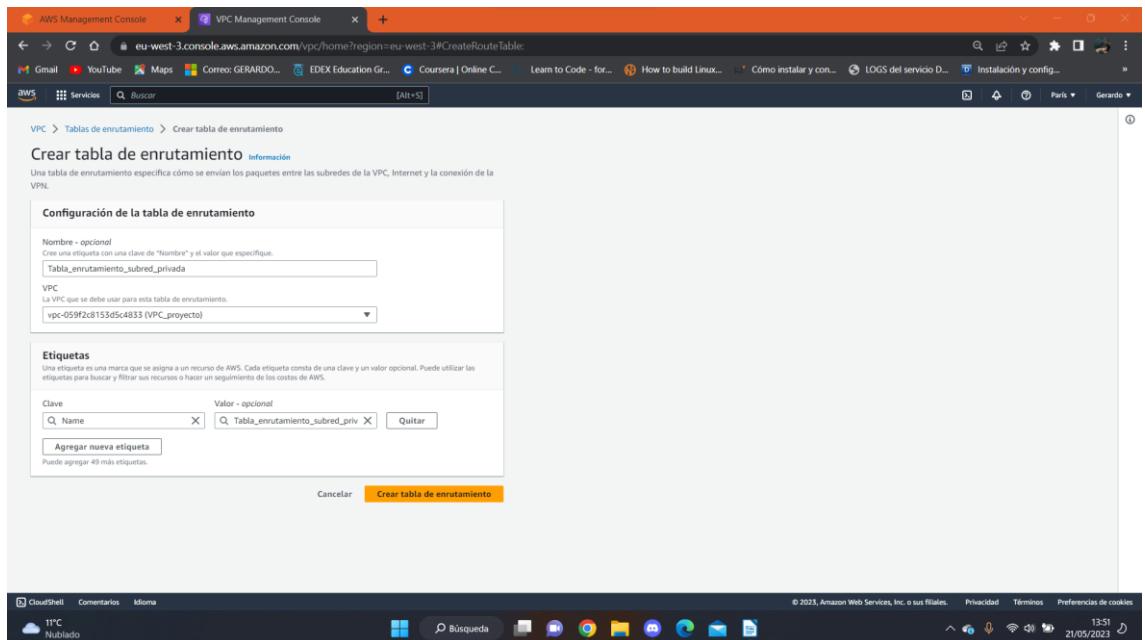
- Para acceder al menú de las tablas de enrutamiento, nos vamos al panel de VPC y seleccionamos 'Tablas de enrutamiento'.



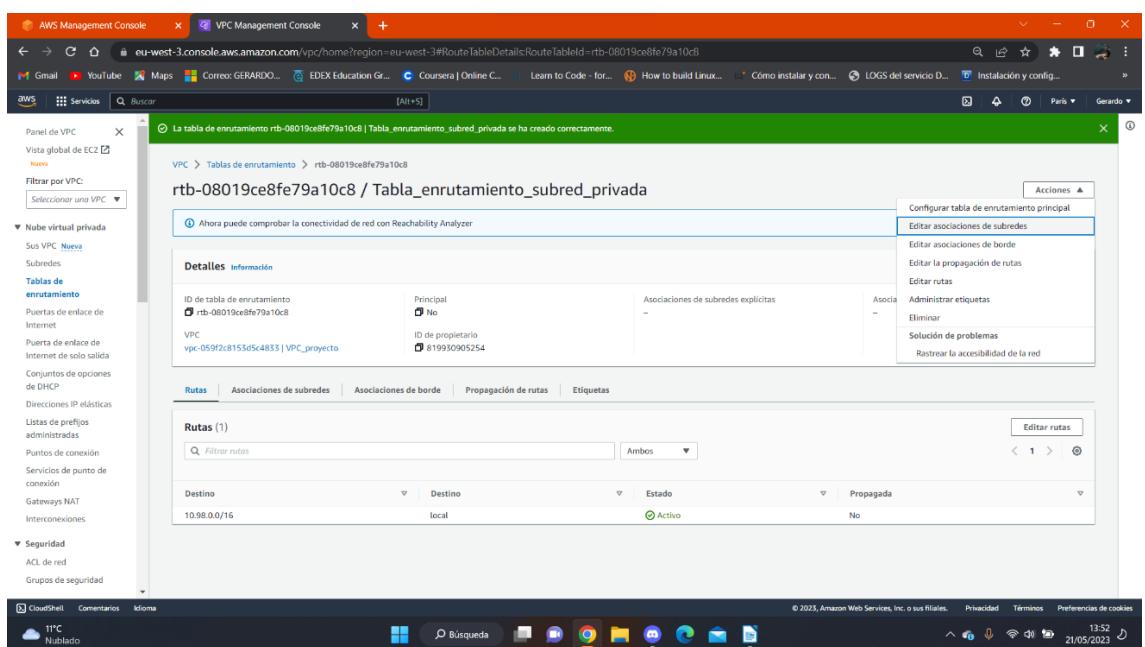
- Vemos como ya existen dos tablas creadas por defecto. Estas tablas se crean con cada VPC, pero no están asignadas a ninguna subred. Las dejamos como están, indicando en su nombre que son las tablas por defecto y no las que vamos a crear ahora. Hacemos click en 'Crear tabla de enrutamiento'.



3. Empezaremos por la tabla de la subred privada. Le asignamos un nombre (Tabla\_enrutamiento\_subred\_privada). En VPC, seleccionamos la VPC del proyecto. Sin más, le damos a ‘Crear tabla de enrutamiento’.



4. En la nueva pantalla, le damos a ‘Acciones’ y seleccionamos ‘Editar asociaciones de subredes’.



## Dockerización de un servidor completo en Cloud.

### 5. Marcamos subred\_privada y guardamos la asociación.

The screenshot shows the AWS Management Console with the VPC Management Console open. The URL is [eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#EditRouteTableSubnetAssociations:RouteTableId=rtb-08019ce8fe79a10c8](https://eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#EditRouteTableSubnetAssociations:RouteTableId=rtb-08019ce8fe79a10c8). The page title is "Editar asociaciones de subredes".  
The main table shows two subnets:

Nombre	ID de subred	CIDR IPv4	CIDR IPv6	ID de tabla de enruteamiento
Subred_privada	subnet-0d4e1ba3caa9464f3	10.98.1.0/24	-	Principal (rtb-08278e89d1e942b59 / Tabla_enr...)
Subred_publica	subnet-09edd04f20e31b7a3	10.98.2.0/24	-	Principal (rtb-08278e89d1e942b59 / Tabla_enr...)

A message at the bottom says: "Has actualizado correctamente las asociaciones de subred para rtb-02ebf9911d189cf3 / Tabla\_enruteamiento\_subred\_publica." (The subnet associations have been updated correctly for rtb-02ebf9911d189cf3 / Tabla\_enruteamiento\_subred\_publica.)

### 6. Una vez creada esta tabla, volvemos al menú de las tablas de enrutamiento y repetimos el proceso para crear la tabla que será asociada a la subred pública, que llevará por nombre "Tabla\_enruteamiento\_subred\_publica". Quedan así creadas las tablas de enrutamiento, listas para su configuración.

The screenshot shows the AWS Management Console with the VPC Management Console open. The URL is [eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#RouteTables:sort=desc:tagName](https://eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#RouteTables:sort=desc:tagName). The page title is "Tablas de enruteamiento (4) | VPC Management Console".  
The table shows four route tables:

Name	ID de tabla de enruteamiento	Asociaciones de subredes explícitas	Asociaciones de b... (Principales)	VPC
Tabla_enruteamiento_subred_publica	rtb-02ebf9911d189cf3	subnet-09edd04f20e31b7a3 / Subred_publica	-	No vpc-059f2x8153d5c4833   VPC_proyecto 819930
Tabla_enruteamiento_subred_privada	rtb-08019ce8fe79a10c8	subnet-0d4e1ba3caa9464f3 / Subred_privada	-	No vpc-059f2x8153d5c4833   VPC_proyecto 819930
Tabla_enruteamiento_proyecto_por_defecto	rtb-08278e89d1e942b59	-	- Sí	vpc-059f2x8153d5c4833   VPC_proyecto 819930
Tabla_enruteamiento_por_defecto	rtb-0226818ab5d6c672b	-	- Sí	vpc-0147945c99a09760c   VPC_por_defecto 819930

A message at the top says: "Ha actualizado correctamente las asociaciones de subred para rtb-02ebf9911d189cf3 / Tabla\_enruteamiento\_subred\_publica." (The subnet associations have been updated correctly for rtb-02ebf9911d189cf3 / Tabla\_enruteamiento\_subred\_publica.)

7. Ahora vamos a configurar las tablas. Empecemos por la asociada a la subred pública.

La tabla de enrutamiento de la subred pública debe permitir el enrutamiento de tráfico de la instancia del bastión con Internet y de la instancia del bastión con la instancia del servidor. La tabla ya trae por defecto una regla que permite el tráfico con toda la VPC.

Para crear rutas, seleccionamos la tabla de enrutamiento y en la pestaña de abajo seleccionamos ‘Rutas’ y seguidamente ‘Editar rutas’.

AWS Management Console

eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#RouteTables.sort=desc:tag:Name

Vista global de EC2

Filtrar por VPC:

Seleccionar una VPC

Nube virtual privada

Sus VPC Nueva Subredes Tablas de enruteamiento Puentas de enlace de Internet Puerta de enlace de Internet de solo salida Conjuntos de opciones de DHCP Direcciones IP elásticas Listas de prefijo administradas Puntos de conexión Servicios de punto de conexión Gateways NAT Interconexiones Seguridad ACL de red Grupos de seguridad

CloudShell Comentarios Mírame

11°C Nublado

eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#RouteTables.sort=desc:tag:Name

Panel de VPC

Información

Find routes by attribute or tag

Acciones Crear tabla de enruteamiento

Name	ID de tabla de enruteamiento	Asociaciones de subredes explícitas	Asociaciones de b... Principales	VPC	ID de...
Tabla_enrutamiento_subred_publica	rtb-02ebf9911d189cf93	subnet-09eddf04f20e51b7a3 / Subred_publica	-	No vpc-059f2c8153dc4833   VPC_proyecto	819930.
Tabla_enrutamiento_subred_privada	rtb-08019ce0f79a10cb	subnet-0d4e1ba3ca59464f / Subred_privada	-	No vpc-059f2c8153dc4833   VPC_proyecto	819930.
Tabla_enrutamiento_proyecto_por_defecto	rtb-08279e891e942b59	-	-	Sí vpc-059f2c8153dc4833   VPC_proyecto	819930.
Tabla_enrutamiento_por_defecto	rtb-022e81a5b9c672b	-	-	Sí vpc-0147845c99a09760c   VPC_por_defecto	819930.

rb-02ebf9911d189cf93 / Tabla\_enrutamiento\_subred\_publica

Detalles Rutas Asociaciones de subredes Asociaciones de borde Propagación de rutas Etiquetas

Rutas (1)

Destino Destino Estado Propagada

10.98.0.0/16 local Activo No

Editar rutas

© 2023, Amazon Web Services, Inc. o sus filiales. Privacidad Términos Preferencias de cookies

13:58 21/05/2023

## Dockerización de un servidor completo en Cloud.

La subred pública necesita las siguientes rutas:

- Una ruta que permite el tráfico IPv4 con Internet. Así, el destino será '0.0.0.0/0' (todos los destinos) y en el segundo campo de destino ponemos el ID de la Internet Gateway, que permite el acceso a Internet.
- Una regla idéntica a la anterior, pero para el tráfico IPv6. A pesar de que no hemos configurado nada relativo a IPv6, vale más cubrirse las espaldas ante posibles cambios de red futuros. Así, el destino de la regla será '::/0' y el segundo destino será de nuevo el ID de la Internet Gateway.
- Una regla que permite el tráfico entre el bastión y el servidor. Esta regla tendrá como destino la subred privada '10.98.1.0/24' y como segundo destino el ID de la instancia del servidor.

The screenshot shows the AWS Management Console with the URL [eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#EditRoutes:RouteTableId=rtb-02ebf9911d189cf93](https://eu-west-3.console.aws.amazon.com/vpc/home?region=eu-west-3#EditRoutes:RouteTableId=rtb-02ebf9911d189cf93). The page title is 'Editar rutas'. The table lists four routes:

Destino	Destino	Estado	Propagada
10.98.0.0/16	Q local	Activo	No
Q 0.0.0.0/0	Q igw-06b146fc501108eb4	-	No <input type="button" value="Quitar"/>
Q ::/0	Q igw-06b146fc501108eb4	-	No <input type="button" value="Quitar"/>
Q 10.98.1.0/24	Q i-0769e1aba7b3acff9	-	No <input type="button" value="Quitar"/>

At the bottom right of the table are buttons for 'Cancelar', 'Vista previa', and a highlighted 'Guardar cambios' button. The browser's address bar shows the full URL. The AWS navigation bar at the top includes 'AWS', 'Servicios', and a search bar. The taskbar at the bottom of the screen shows various open applications like CloudShell, Comments, and Idioms, along with system icons for battery, signal, and date/time.

8. Vamos ahora con la configuración de la tabla asociada a la red privada, que debe permitir el enrutamiento de tráfico de la instancia del servidor con la instancia del bastión y, como el bastión actuará como servidor NAT para proporcionarle salida a internet al servidor, también debe permitir el tráfico del servidor con internet a través de la Gateway NAT. Como antes, la tabla incluye una regla predeterminada que permite el tráfico VPC. Estas serían las reglas que habría que añadir:

- Una regla que permita el tráfico entre la instancia del servidor y la instancia del bastión. El destino será la subred pública '10.98.2.0/24' y el siguiente destino el ID de la instancia del bastión.
- Una regla que permita el tráfico entre el servidor e internet a través de la Gateway NAT creada para este propósito. El destino será '0.0.0.0/0' (todos los destinos) y el siguiente destino será el ID de la Gateway NAT.

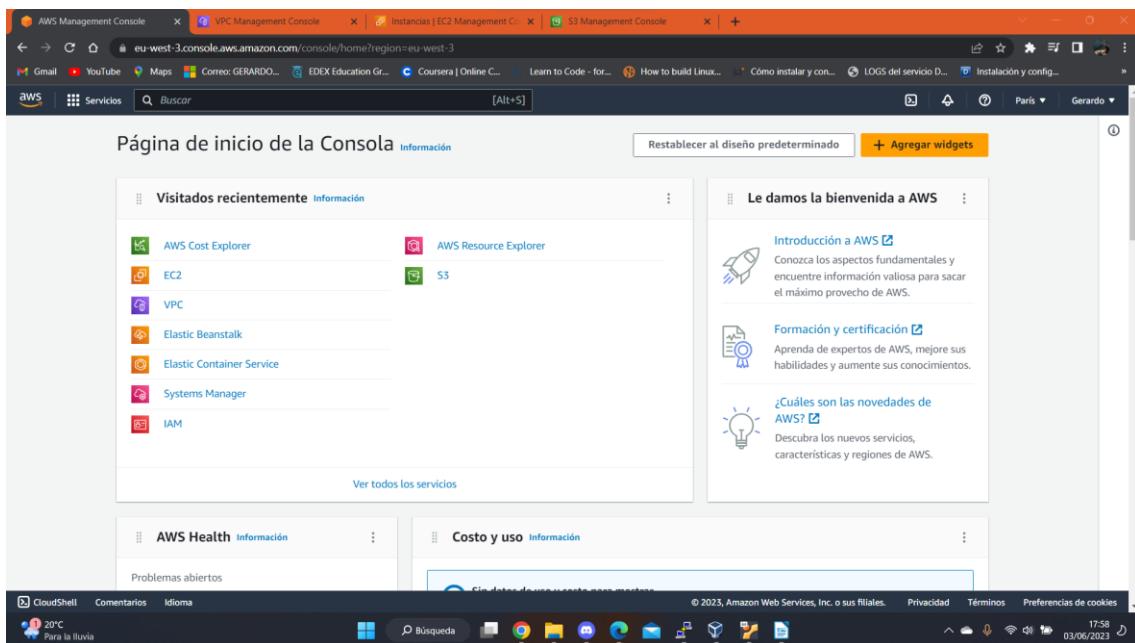
Con esto quedan configuradas las tablas de enruteamiento y finalizada la creación de la infraestructura, con la excepción de los grupos de seguridad, que se configurarán más adelante. Pulsa [aquí](#) para volver arriba.

## 10.4 Anexo 4: Manual de creación y uso del Bucket S3.

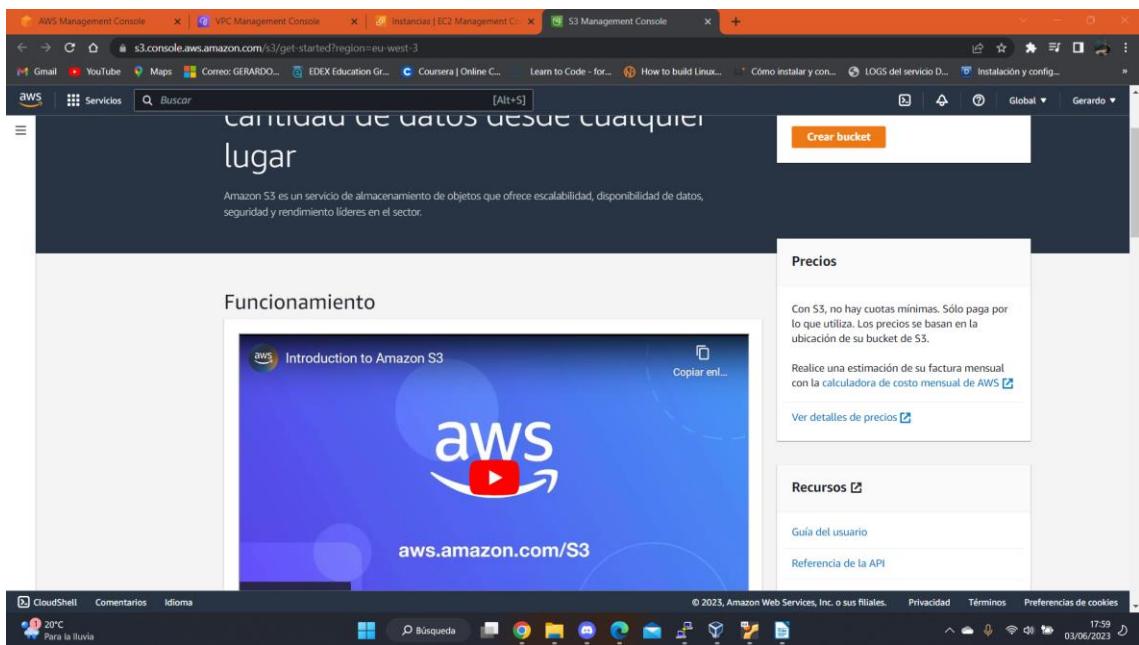
Cada bucket de S3 tiene un nombre único a nivel global y se accede a través de una URL única. Los buckets de S3 son altamente escalables y pueden almacenar una cantidad prácticamente ilimitada de datos (si pagas el servicio). Además, proporcionan alta disponibilidad y durabilidad de los datos. Este manual también incluye la instalación del cliente de línea de comandos (CLI) de AWS, necesario para gestionar el bucket desde Ubuntu y la creación de un usuario AIM con unas claves para poder acceder al bucket.

Vamos a ver el proceso de creación y utilización de uno de estos buckets:

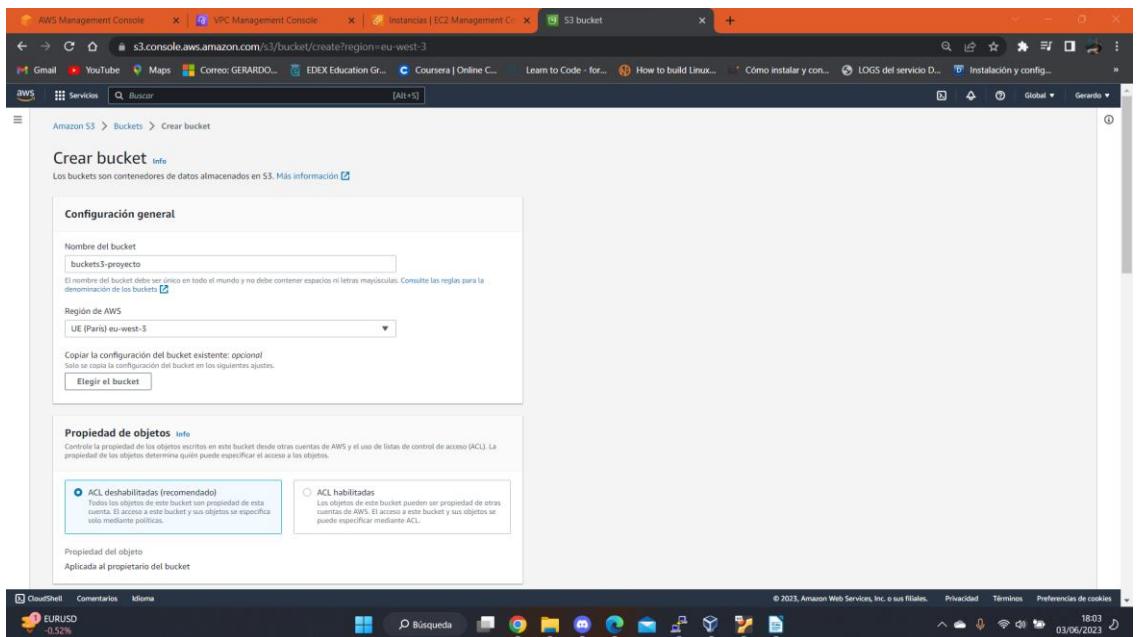
1. En la página de inicio de la consola de AWS, buscamos ‘S3’ en la barra de búsqueda. Si lo tenemos entre los servicios visitados recientemente, pulsamos en ‘S3’



2. En la nueva ventana, damos a ‘Crear Bucket’.

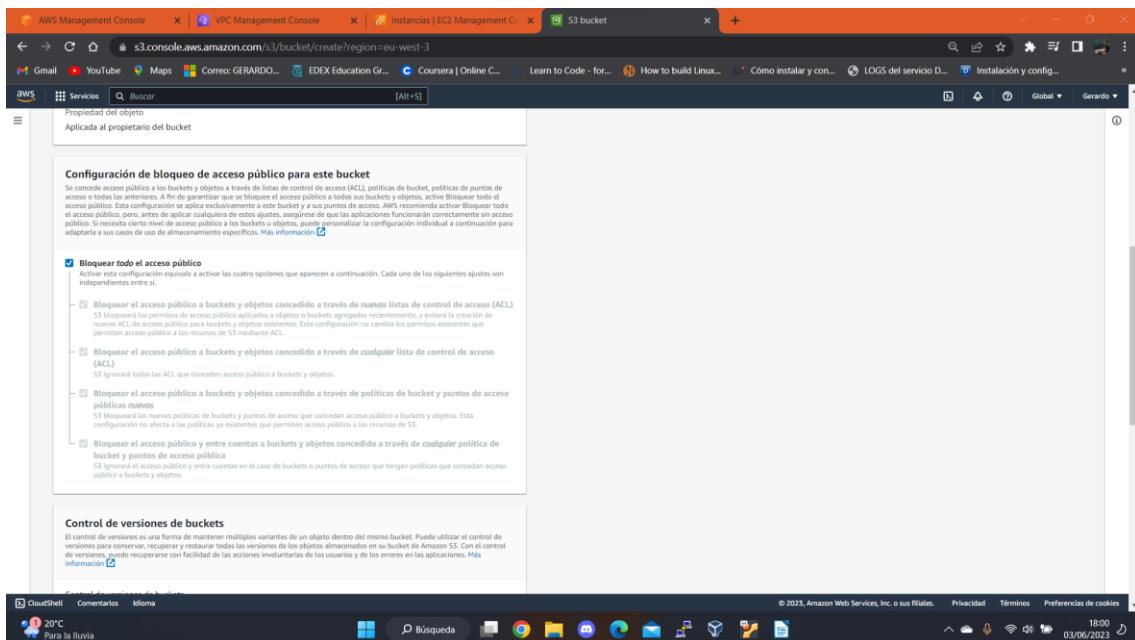


3. En el menú de creación de buckets, introducimos un nombre para el mismo. Debemos tener en cuenta que los nombres de los buckets deben ser único, por lo que tendremos que elegir uno que no esté ya en uso. Seleccionamos la región de AWS en la que queremos que se localice el recurso. Como no necesitamos control de usuarios, seleccionamos ‘ACL deshabilitadas’.



## Dockerización de un servidor completo en Cloud.

En la configuración de bloqueo de acceso público dejamos marcado ‘Bloquear todo’.



**Configuración de bloqueo de acceso público para este bucket**

Se concede acceso público a los buckets y objetos a través de listas de control de acceso (ACL), políticas de bucket, políticas de objetos de acceso público y control de versiones. Si activa el control de acceso público, active Bloquear todo el acceso público. Esta configuración se aplica exclusivamente a este bucket y a sus puntos de acceso. AWS recomienda activar Bloquear todo el acceso público, pero, antes de aplicar cualquiera de estos ajustes, asegúrese de que las aplicaciones funcionarán correctamente sin acceso público. Si necesita acceso público a los buckets y objetos, consulte la documentación y personalice la configuración individualizada para adaptarla a sus casos de uso de almacenamiento específicos. [Más información](#)

Bloquear todo el acceso público

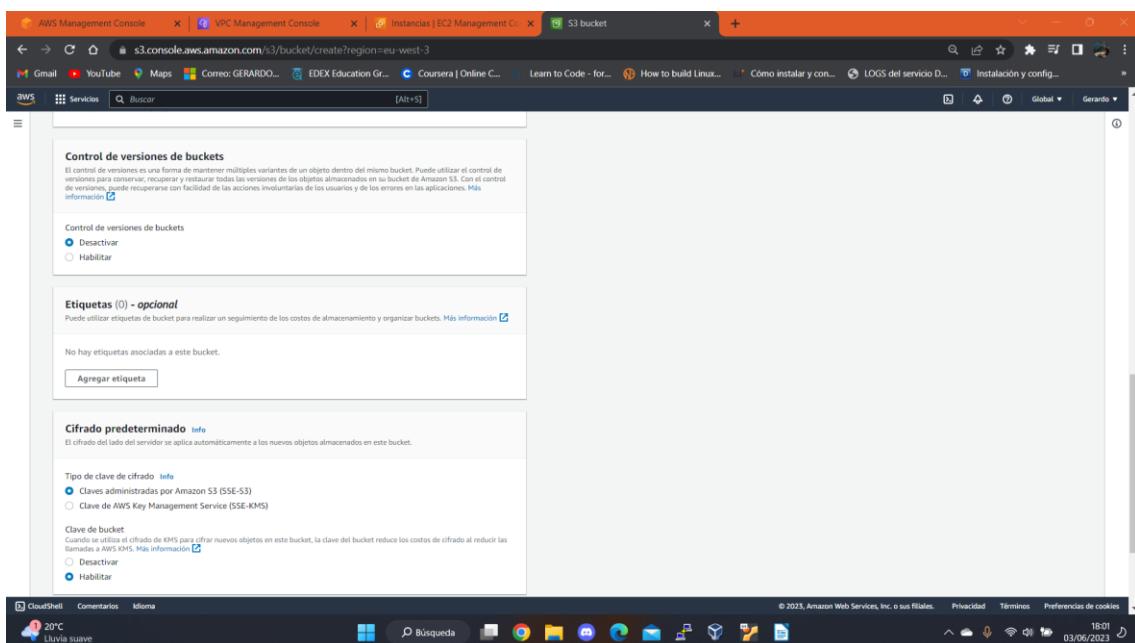
Activar esta configuración equivale a activar las cuatro opciones que aparecen a continuación. Cada uno de los siguientes ajustes son independientes entre sí.

- Bloquear el acceso público a buckets y objetos concedido a través de nuevas listas de control de acceso (ACL)  
S3 bloqueará los permisos de acceso público aplicados a objetos o buckets agregados recientemente, y evitará la creación de nuevas listas de acceso público para buckets y objetos existentes. Esta configuración no cambia los permisos existentes que permiten acceso público a los buckets y objetos.
- Bloquear el acceso público a buckets y objetos concedido a través de cualquier lista de control de acceso (ACL)  
S3 ignorará todas las ACL que conceden acceso público a buckets y objetos.
- Bloquear el acceso público a buckets y objetos concedido a través de políticas de bucket y puntos de acceso públicos nuevos  
S3 bloqueará las nuevas políticas de buckets y puntos de acceso que conceden acceso público a buckets y objetos. Esta configuración no afecta a las políticas ya existentes que permiten acceso público a los recursos de S3.
- Bloquear el acceso público y entre cuentas a buckets y objetos concedido a través de cualquier política de bucket y puntos de acceso pública  
S3 ignorará el acceso público y entre cuentas en el caso de buckets o puntos de acceso que tengan políticas que concedan acceso público a buckets y objetos.

**Control de versiones de buckets**

El control de versiones es una forma de mantener múltiples variantes de un objeto dentro del mismo bucket. Puede utilizar el control de versiones para conservar, recuperar y restaurar todas las versiones de los objetos almacenados en su bucket de Amazon S3. Con el control de versiones, puede recuperarse con facilidad de las acciones involuntarias de los usuarios y de los errores en las aplicaciones. [Más información](#)

En control de versiones marcamos ‘Desactivar’. En lo referente a claves, dejamos las opciones por defecto.



**Control de versiones de buckets**

El control de versiones es una forma de mantener múltiples variantes de un objeto dentro del mismo bucket. Puede utilizar el control de versiones para conservar, recuperar y restaurar todas las versiones de los objetos almacenados en su bucket de Amazon S3. Con el control de versiones, puede recuperarse con facilidad de las acciones involuntarias de los usuarios y de los errores en las aplicaciones. [Más información](#)

Control de versiones de buckets

Desactivar

Habilitar

**Etiquetas (0) - opcional**

Puede utilizar etiquetas de bucket para realizar un seguimiento de los costos de almacenamiento y organizar buckets. [Más información](#)

No hay etiquetas asociadas a este bucket.

[Agregar etiqueta](#)

**Cifrado predeterminado** [Info](#)

El cifrado del lado del servidor se aplica automáticamente a los nuevos objetos almacenados en este bucket.

Tipo de clave de cifrado [Info](#)

Claves administradas por Amazon S3 (SSE-S3)

Clave de AWS Key Management Service (SSE-KMS)

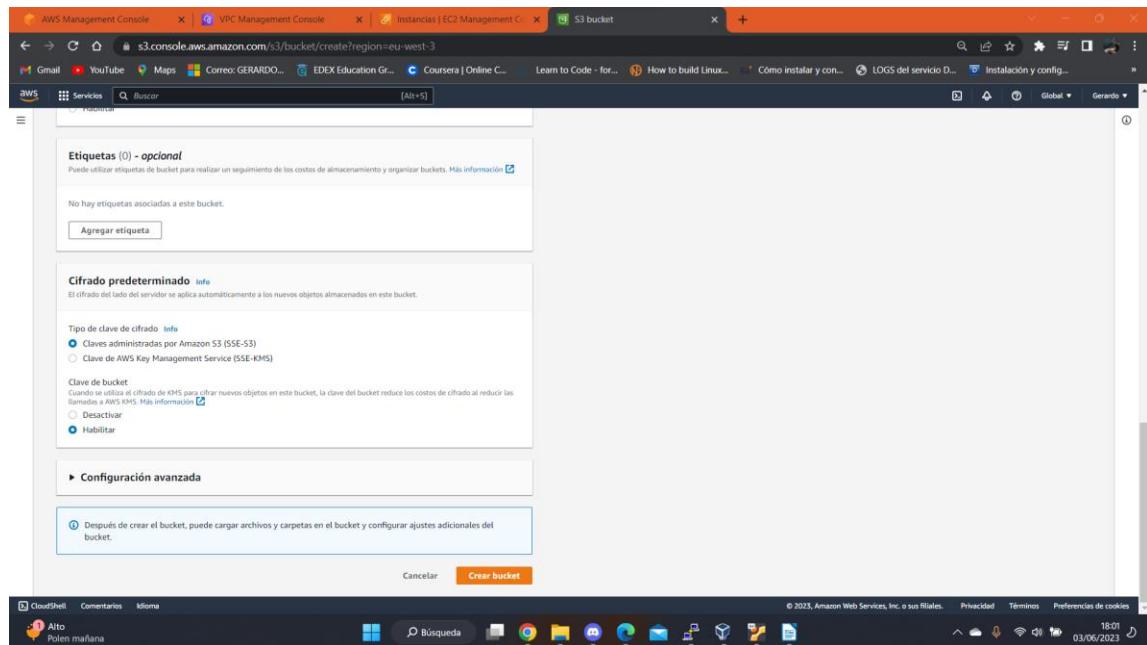
Clave de bucket

Guarda se utiliza el cifrado de KMS para cifrar nuevos objetos en este bucket, la clave del bucket reduce los costos de cifrado al reducir las llamadas a KMS. [Más información](#)

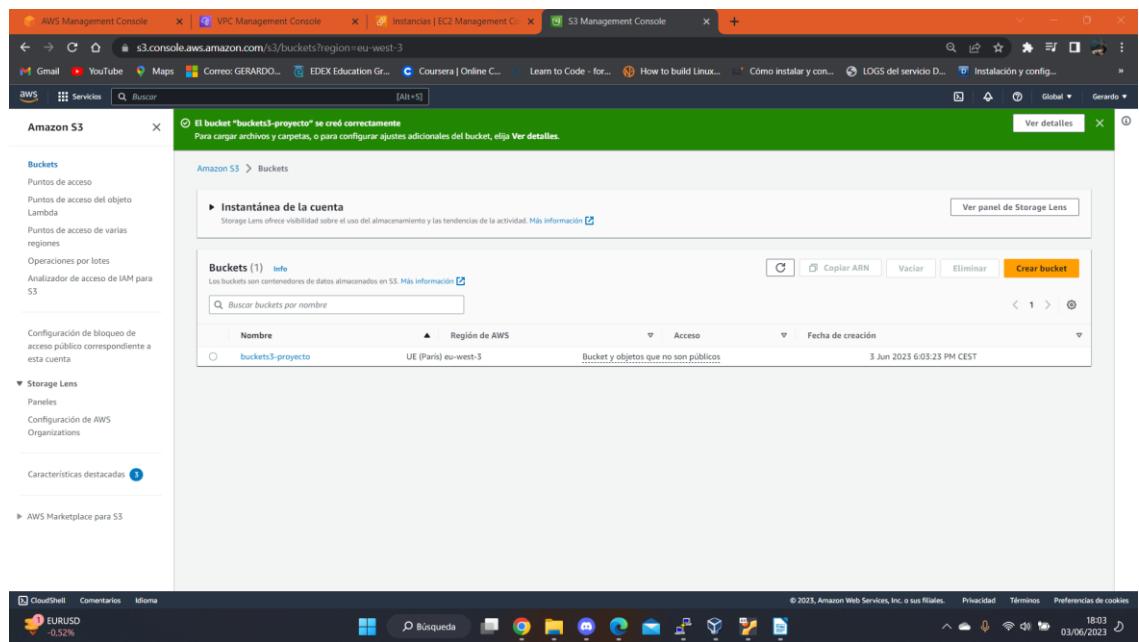
Desactivar

Habilitar

Finalmente, hacemos click en ‘Crear bucket’.

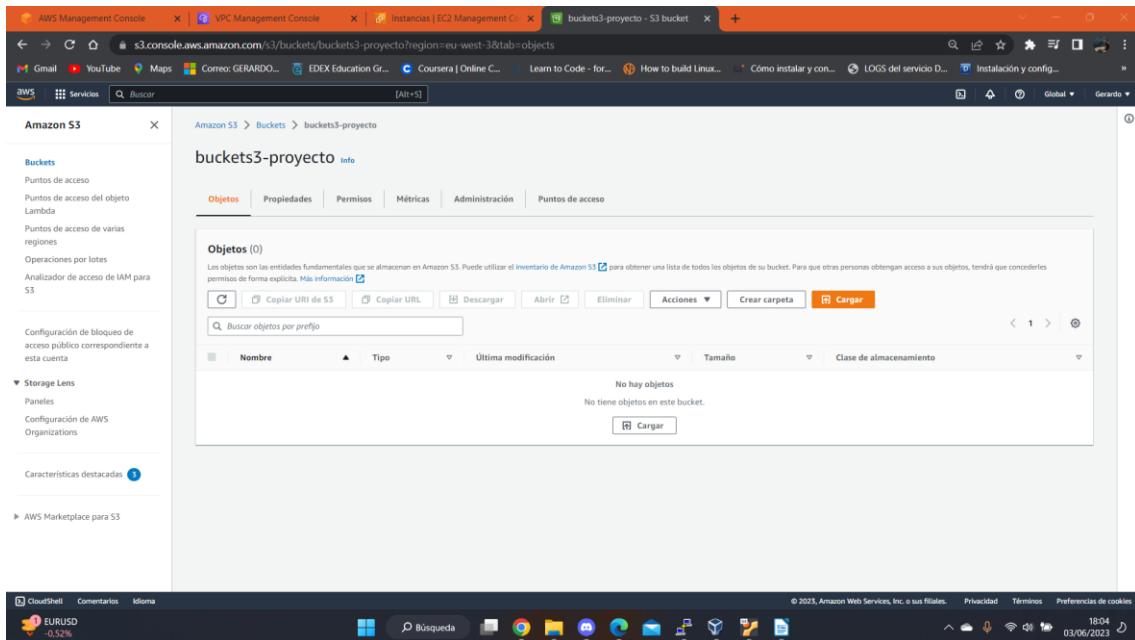


- Pasaremos a la página donde se listan los buckets creados, donde veremos el nuestro. Para subir archivos, pulsaremos en el nombre del bucket.

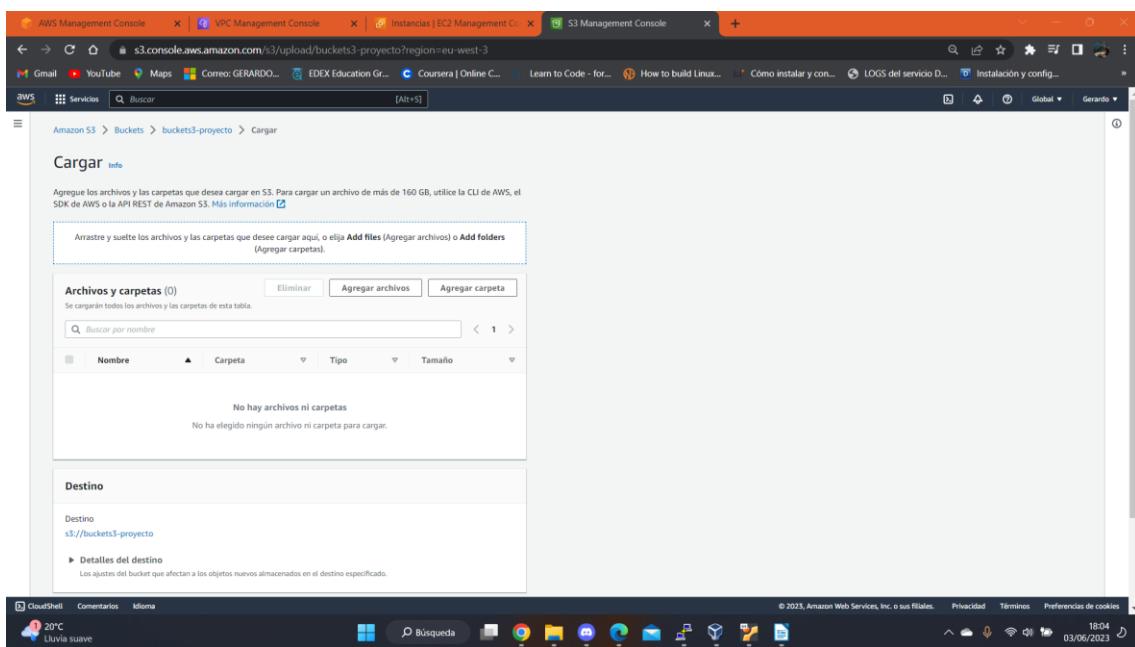


## Dockerización de un servidor completo en Cloud.

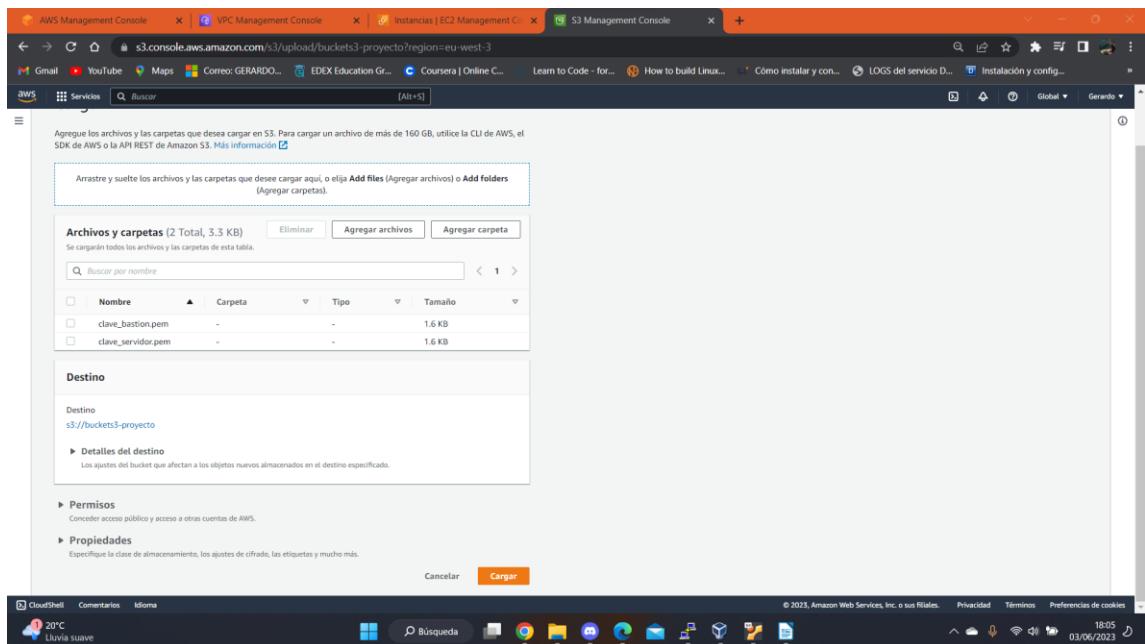
5. Veremos el menú de subida de archivos. En él, le daremos a ‘Cargar’ para comenzar el proceso de subida.



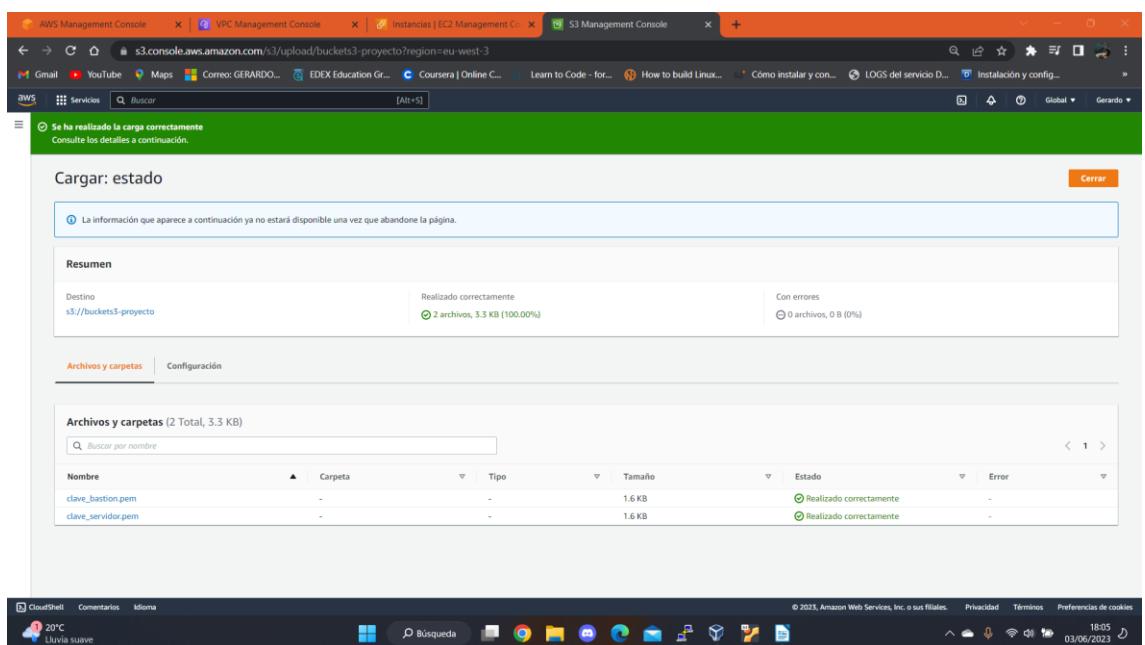
6. Pulsamos en ‘Agregar archivos’ o ‘Agregar carpeta’, según queramos cargar unos u otras. Después, seleccionaremos lo que queremos subir al bucket.



### 7. Una vez seleccionados los archivos, le damos a 'Cargar'.



### 8. Se nos presentará una pantalla que mostrará el estado satisfactorio de la carga.

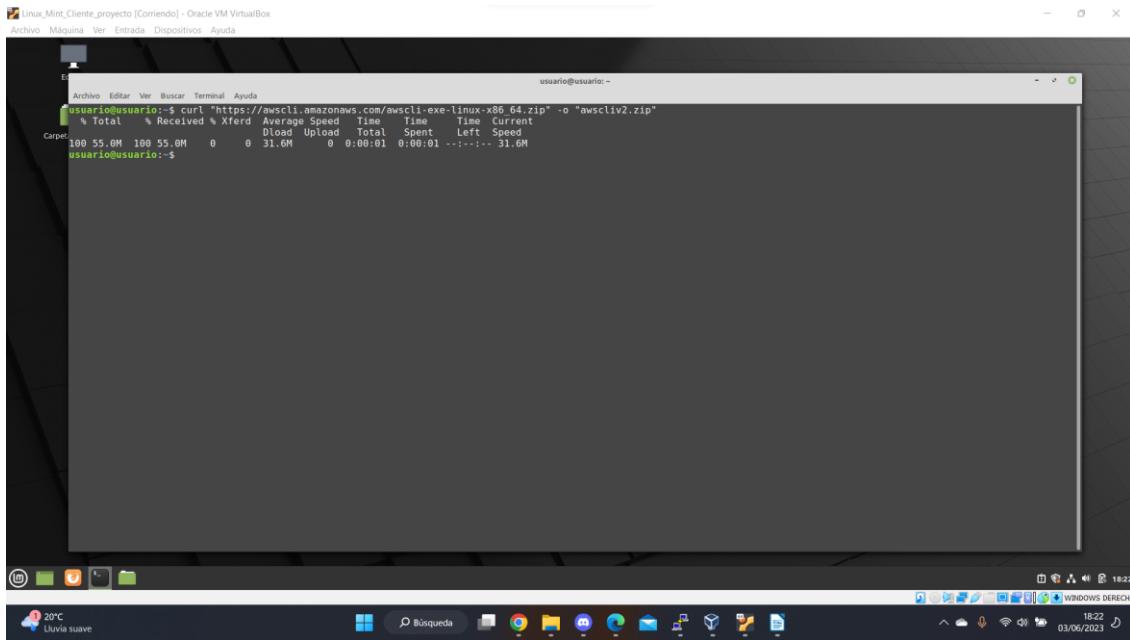


## Dockerización de un servidor completo en Cloud.

Vamos a pasar al uso del bucket desde un entorno Ubuntu 22.04. Para poder utilizarlo, primero debemos realizar la instalación del AWS CLI. Seguimos estos pasos:

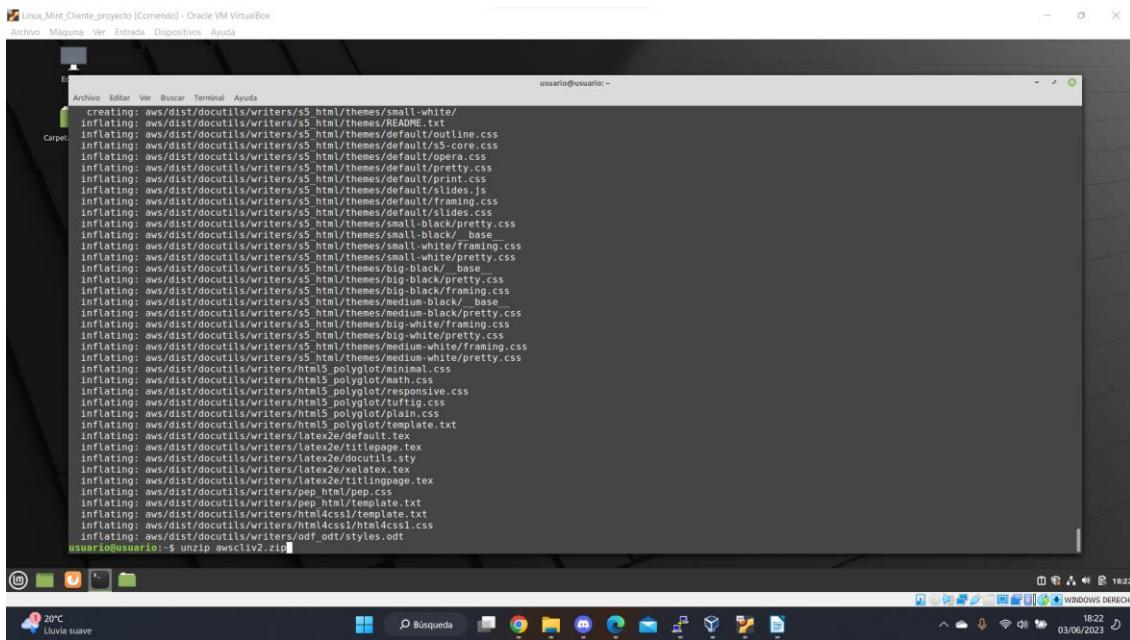
1. Descargamos el archivo ZIP del AWS CLI para poder realizar la instalación después.

- `curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`



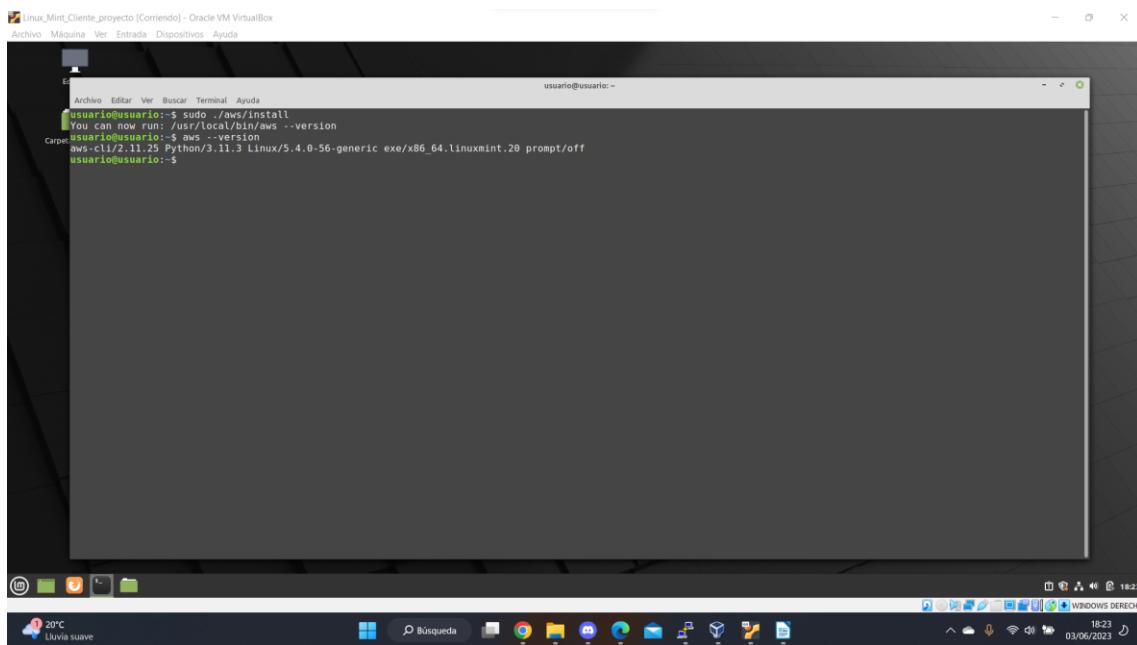
2. Descomprimimos el archivo zip.

- `unzip awscliv2.zip`



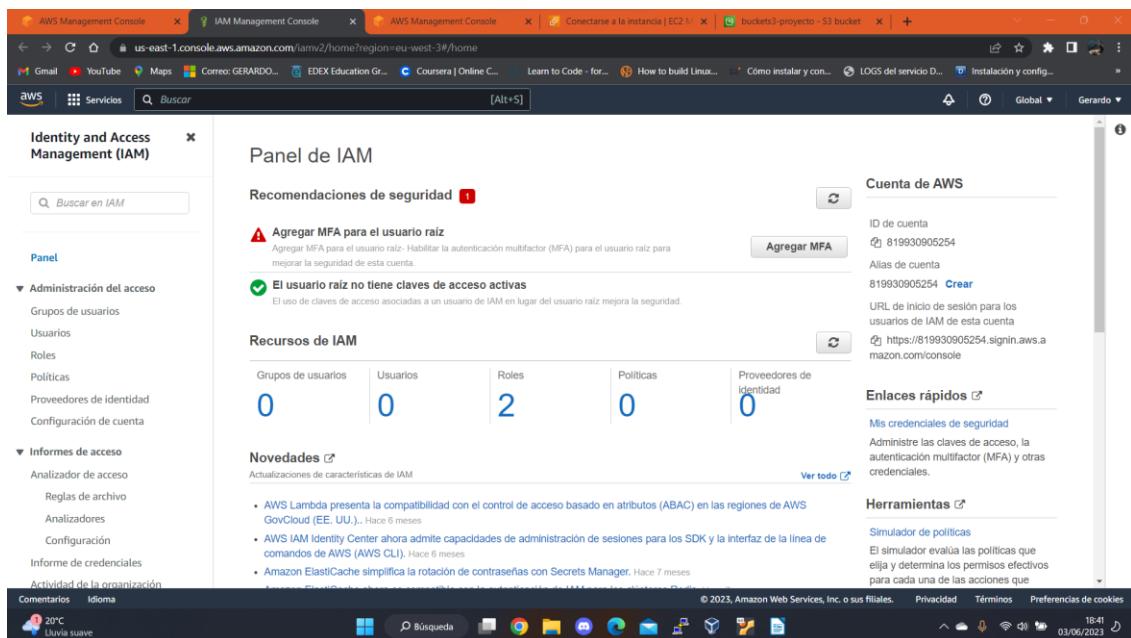
3. Instalamos AWS CLI y comprobamos que está instalado consultando su versión.

- `sudo ./aws/install`
- `aws --version`



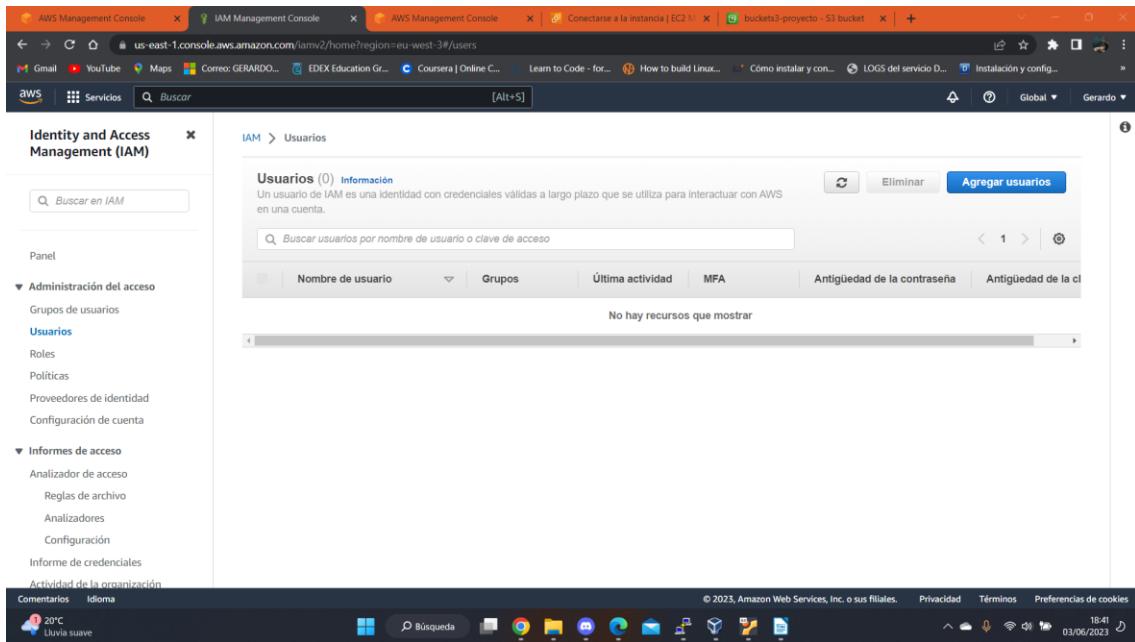
Para acceder al contenido del bucket fuera del entorno AWS, es necesario poseer unas credenciales. La forma más sencilla de obtenerlas es la creación de un usuario IAM. Vamos a ver cómo a hacerlo:

1. Desde el panel de la consola AWS, buscamos 'IAM'. Una vez en el panel de IAM, hacemos click en 'Usuarios'

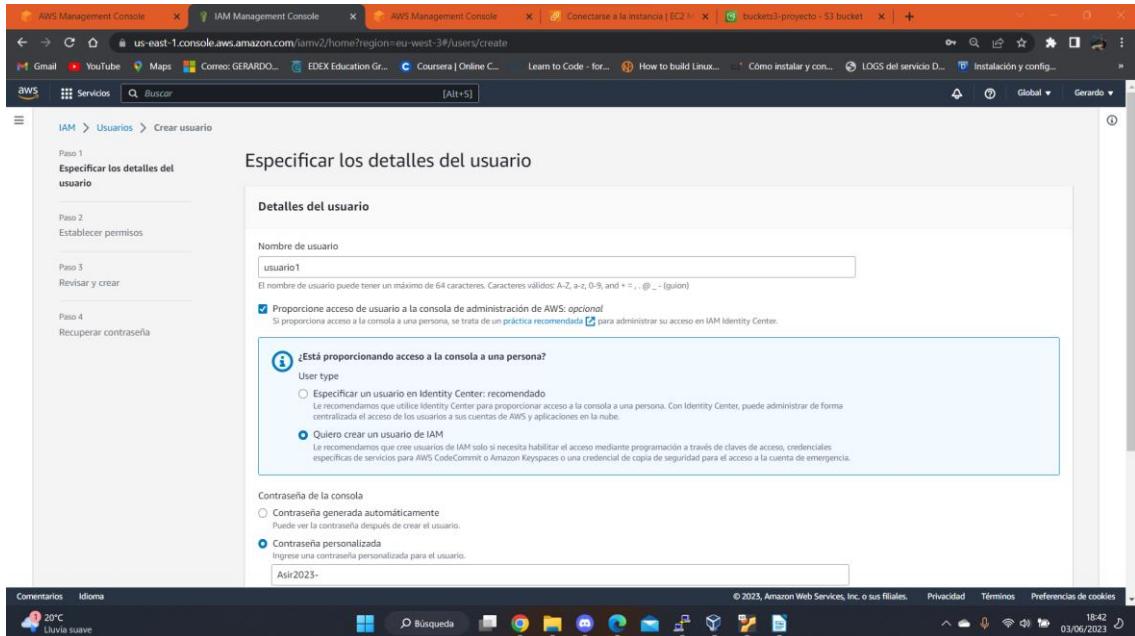


## Dockerización de un servidor completo en Cloud.

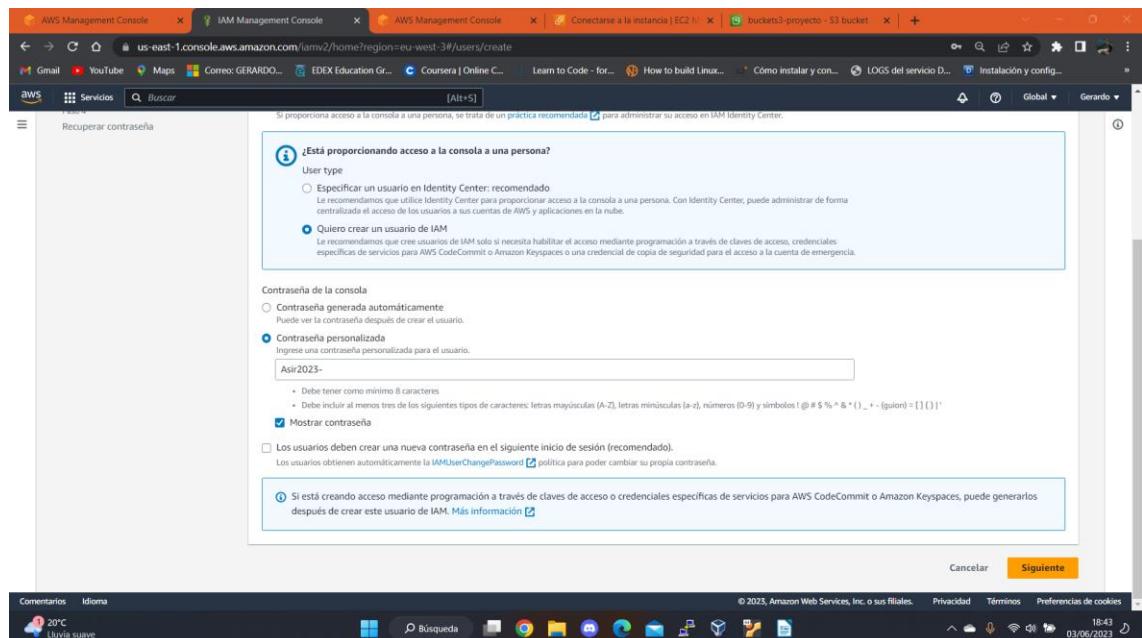
2. En la pantalla usuarios hacemos click en ‘Agregar usuarios’.



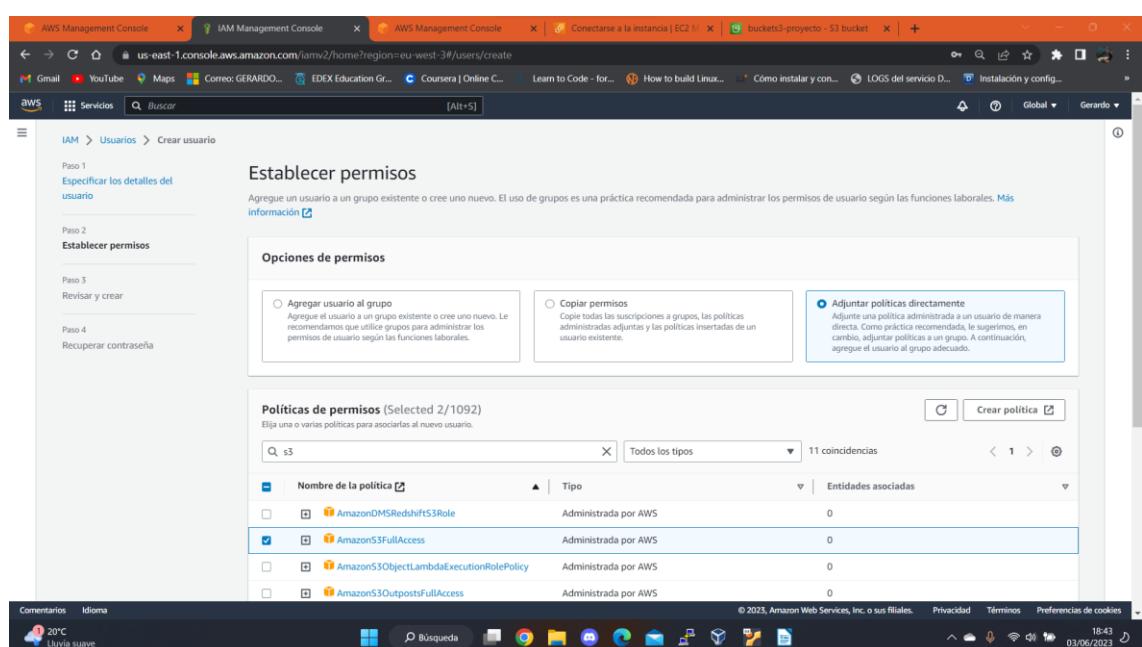
3. Introducimos el nombre de usuario y marcamos ‘Proporcione acceso de usuario a la consola’. Debajo, seleccionamos ‘Quiero crear un usuario de IAM’. A continuación, elegimos ‘Contraseña personalizada’ y la introducimos.



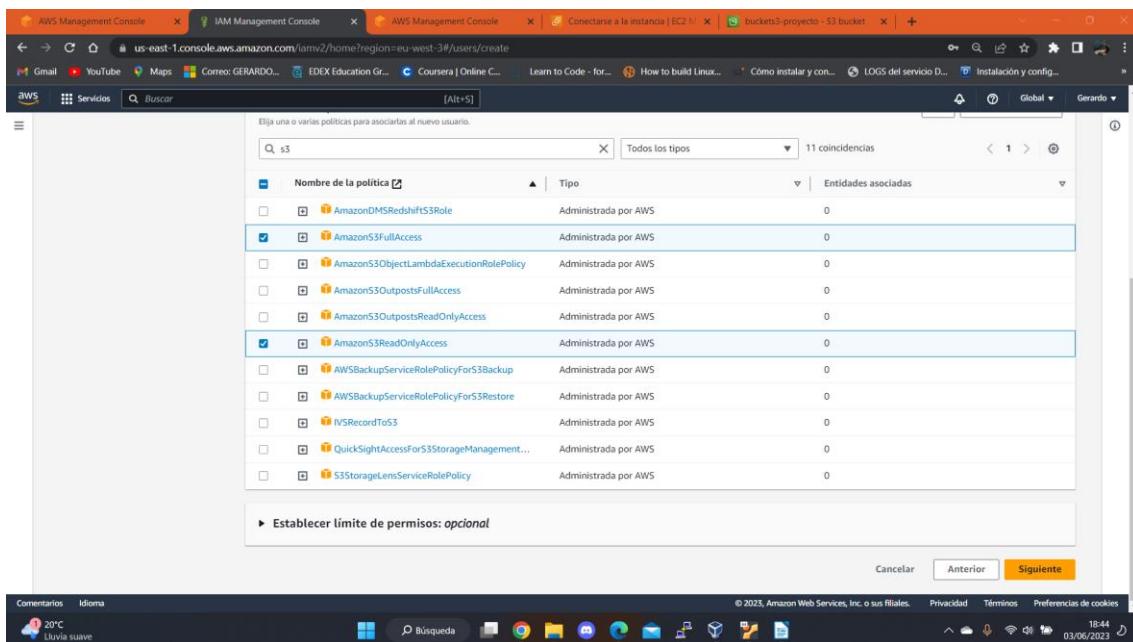
Desmarcamos la opción ‘Los usuarios deben crear una nueva contraseña’ y pulsamos en ‘Siguiente’



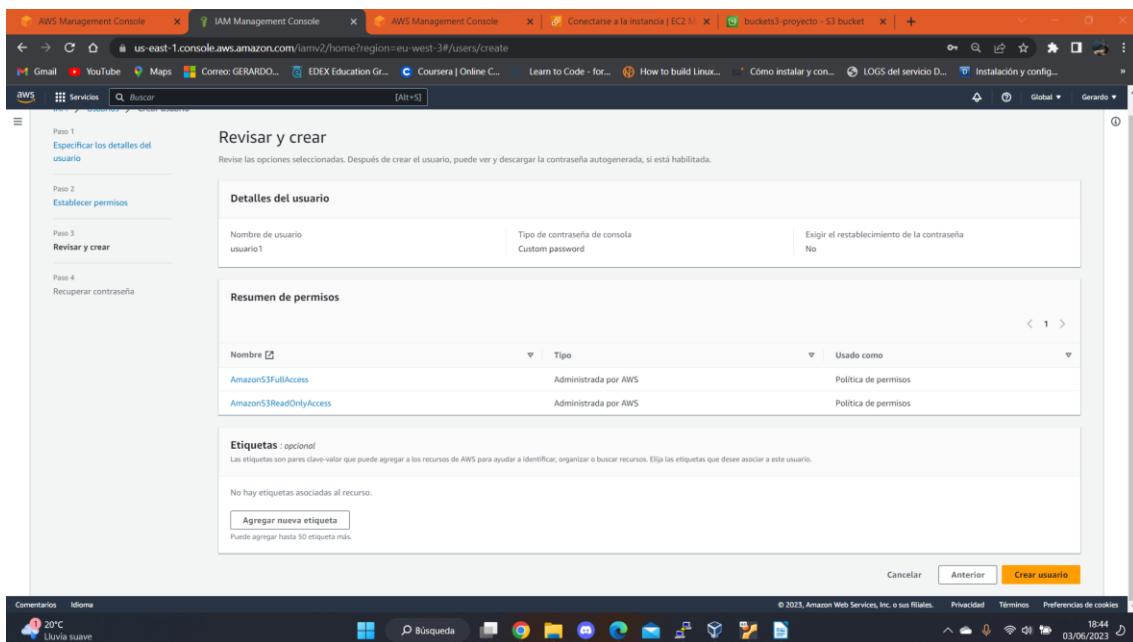
- En ‘Establecer permisos’ seleccionamos ‘Adjuntar políticas directamente’. En la barra de búsqueda introducimos ‘s3’ y seleccionamos ‘AmazonS3FullAccess’ y ‘AmazonS3ReadOnlyAccess’. Pulsamos en siguiente.



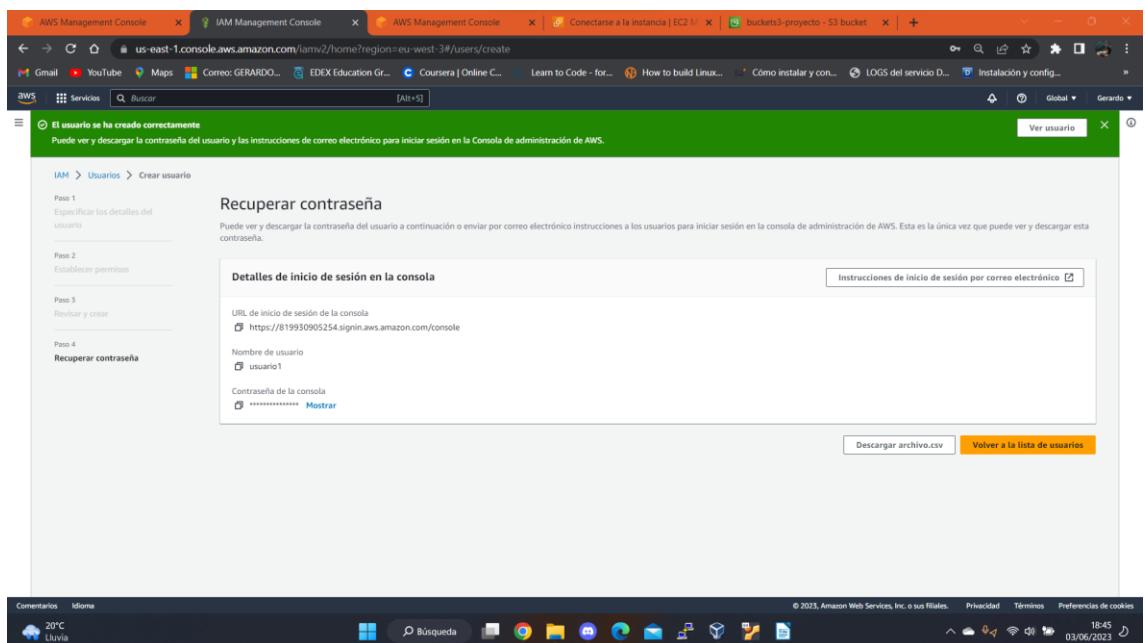
## Dockerización de un servidor completo en Cloud.



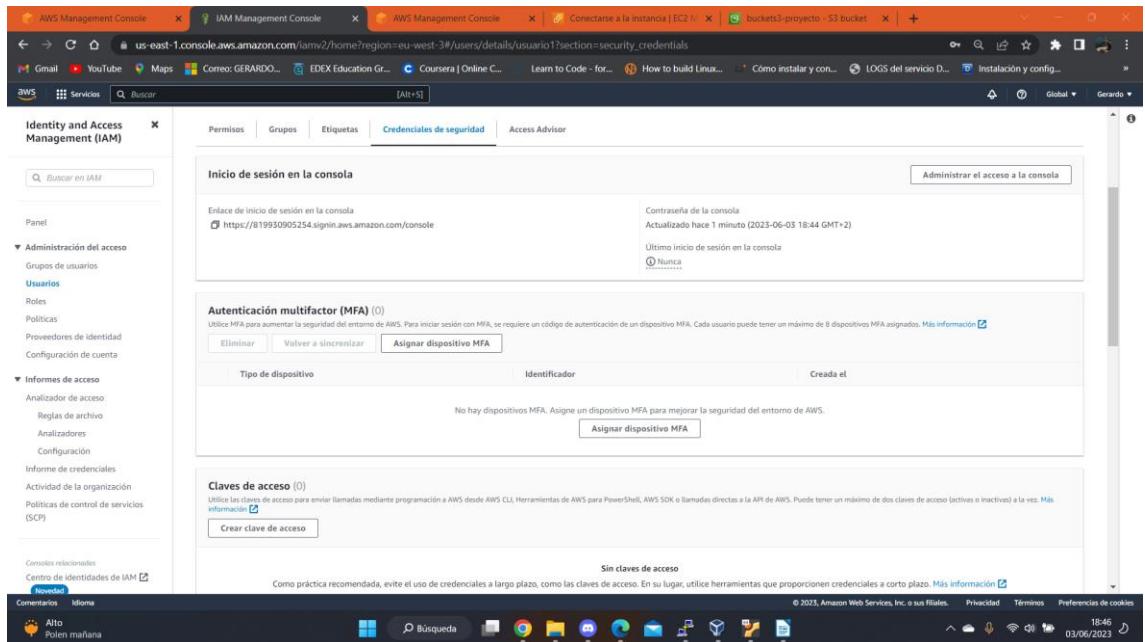
5. Finalmente, pulsamos en ‘Crear usuario’.



6. Es recomendable descargar el archivo.csv que contiene los datos del usuario pulsando en ‘Descargar archivo.csv’. Después hacemos click en ‘Ver usuario’.

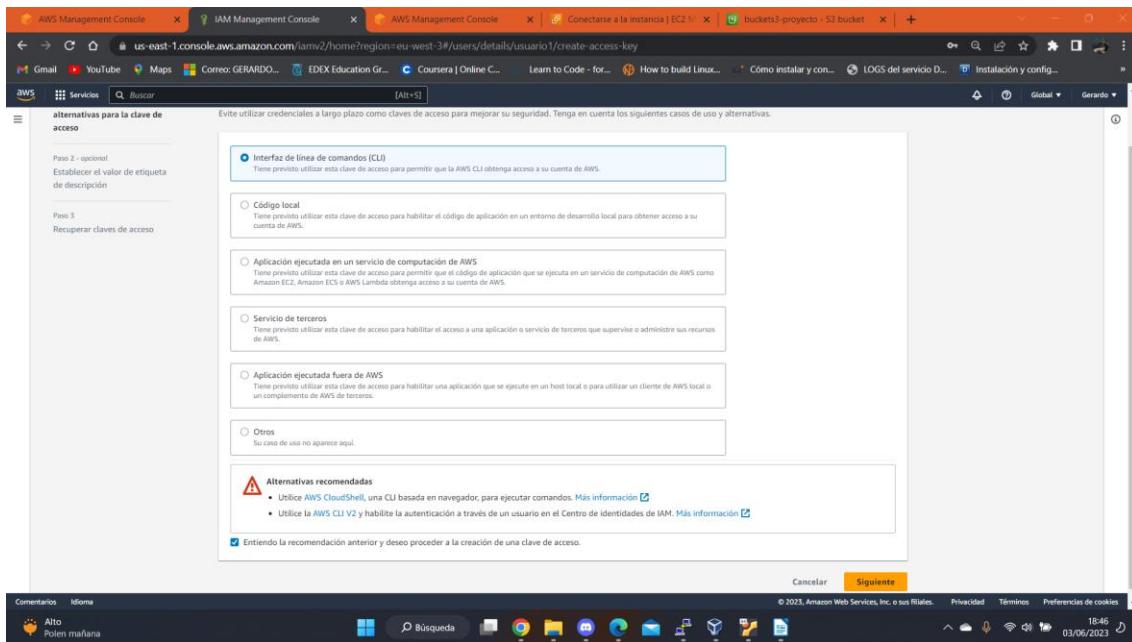


7. Hacemos click en la pestaña ‘Credenciales de seguridad’. Luego, en ‘Crear clave de acceso’.

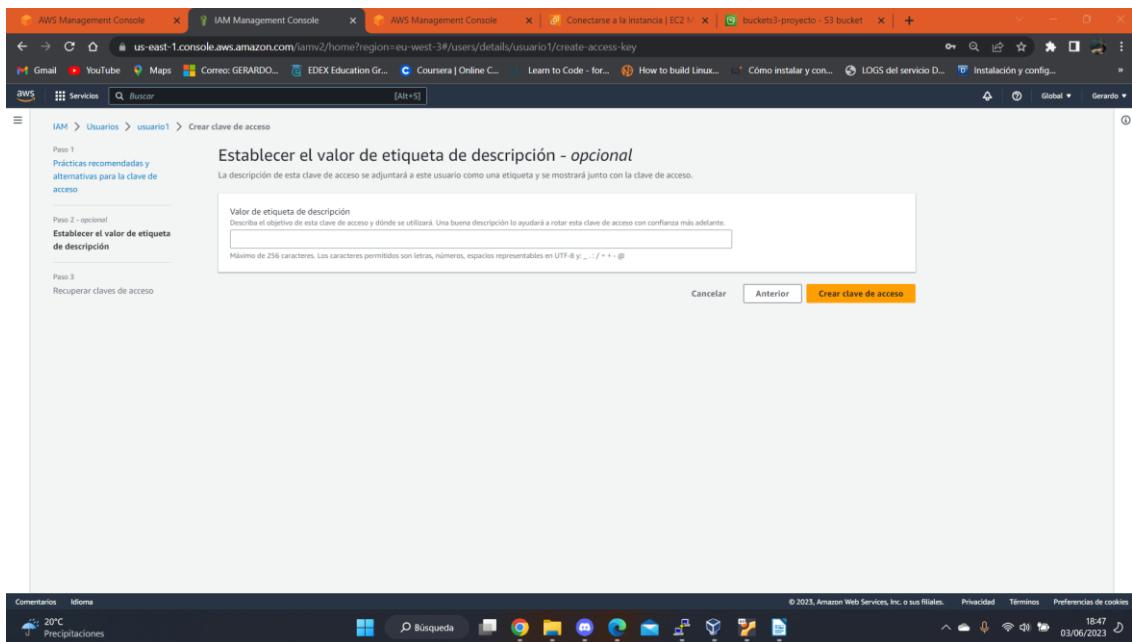


## Dockerización de un servidor completo en Cloud.

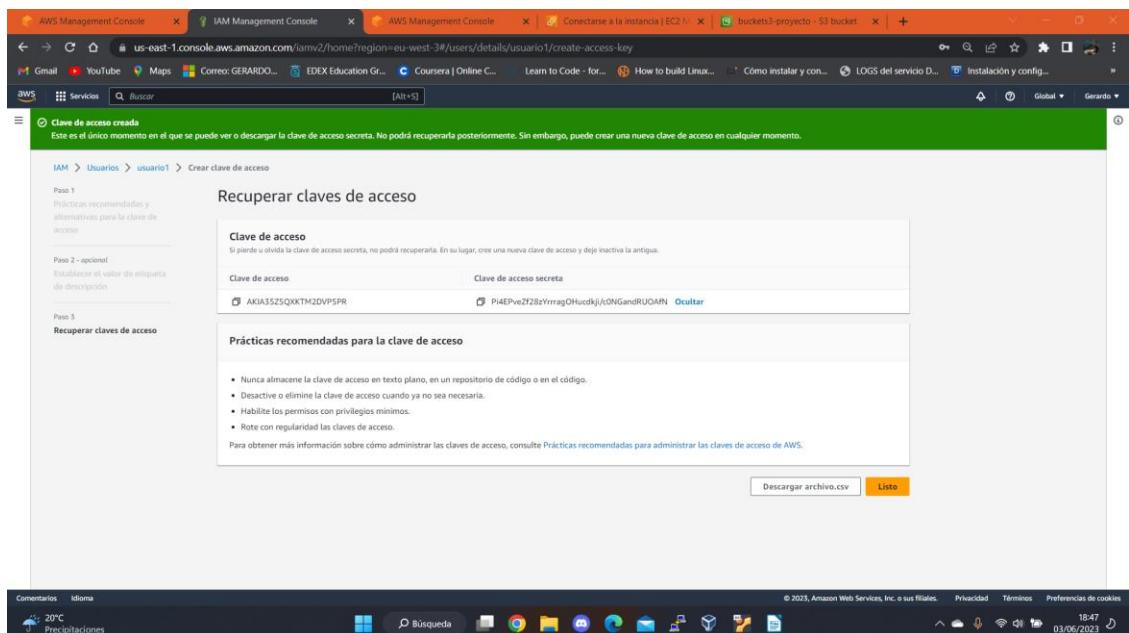
8. Seleccionamos ‘Interfaz de línea de comandos (CLI)’. Después, marcamos ‘Entiendo la recomendación...’ y damos a ‘Siguiente’.



9. Hacemos click en ‘Crear clave de acceso’.

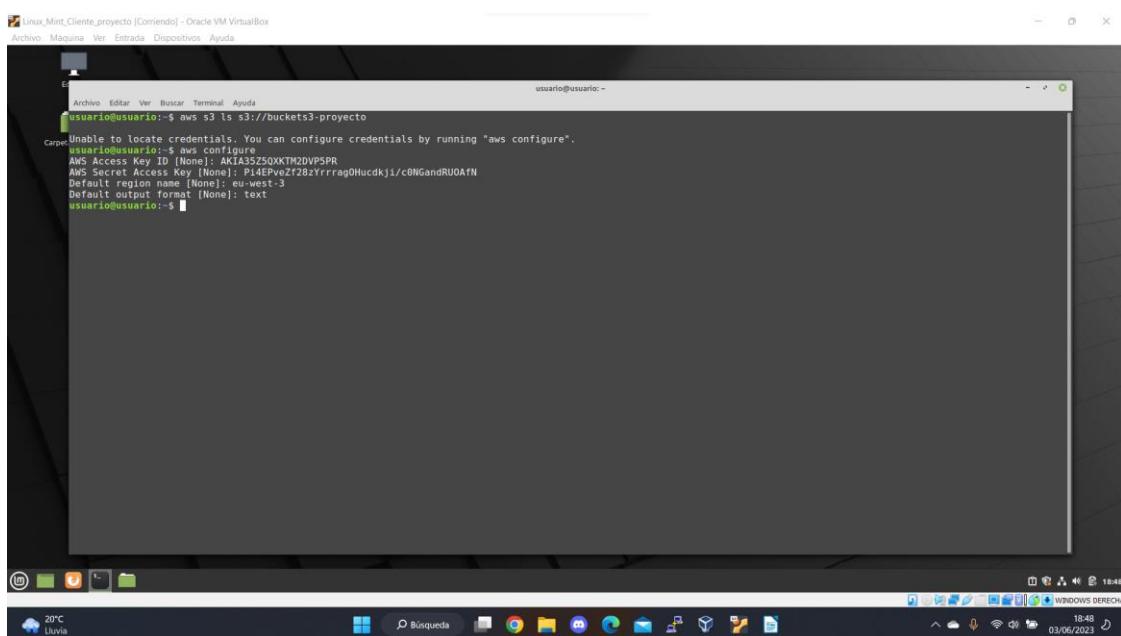


- Tras este laborioso proceso, tendremos por fin en nuestra mano las claves de acceso al bucket. Se recomienda encarecidamente guardarlas en un lugar seguro, descargando el archivo.csv o apuntándolas



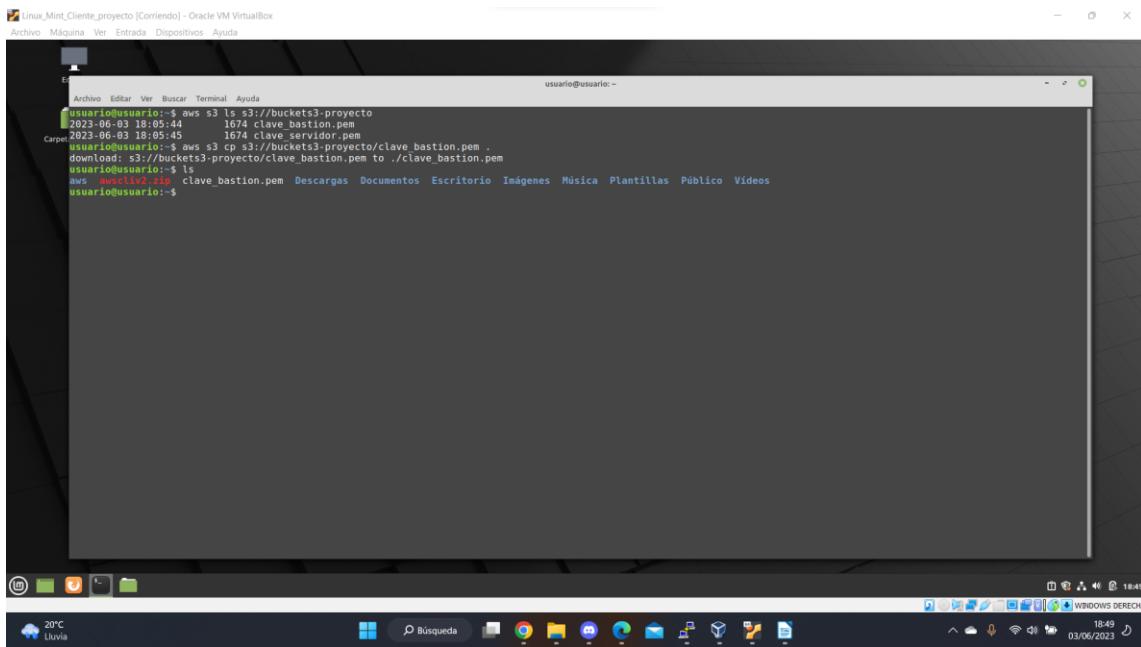
Vamos a terminar este manual con la descarga de un archivo en un cliente.

- Abrimos la consola del cliente y configuraremos la línea de comandos de AWS. Para ello utilizaremos las claves que acabamos de obtener. También indicamos la región de AWS que usamos y el formato de output.
  - aws configure*



2. Por último, listamos el contenido del bucket y nos descargamos un archivo.

- `aws s3 ls s3://buckets3-proyecto`
- `aws s3 cp s3://buckets3-proyecto/clave_bastion.pem`

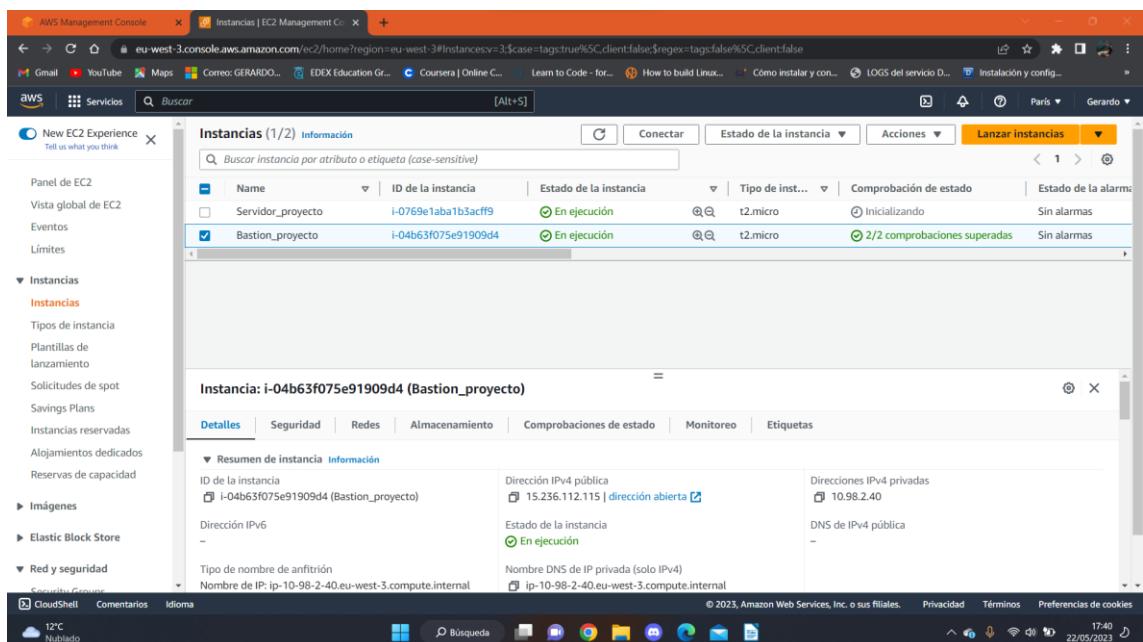


Pulsa [aquí](#) para volver arriba.

## 10.5 Anexo 5: Manual de instalación del proxy inverso.

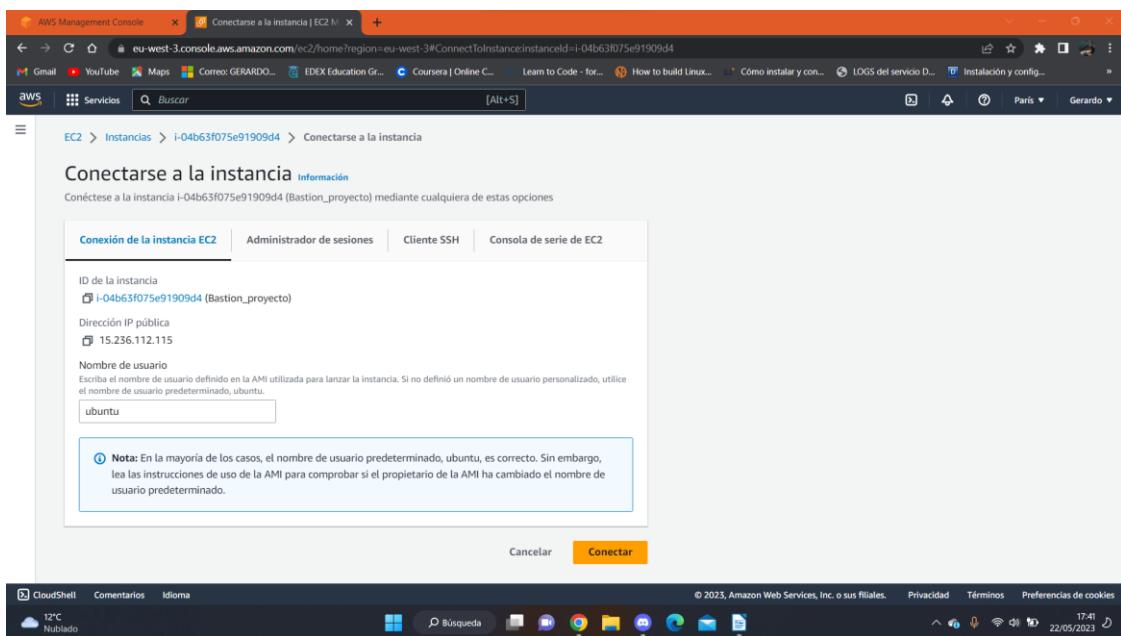
Como decíamos, para el correcto funcionamiento de la infraestructura propuesta se necesita que el bastión reenvíe las peticiones que recibe por parte de los clientes al servidor. Conseguimos esto haciendo uso de un proxy inverso. Un proxy inverso recibe las solicitudes del cliente y las reenvía al servidor, permitiendo ocultar la infraestructura interna al cliente y, así, proporcionando un nivel extra de seguridad y control en el acceso a los servicios. El cliente se comunica exclusivamente con el bastión, y es éste el que controla las solicitudes y redirige el tráfico. Para este propósito, hemos utilizado Nginx, un servidor web/proxy inverso ligero de alto rendimiento. Vamos a describir el proceso de instalación.

Nos conectamos a la instancia bastión desde AWS. Para ello, nos colocamos en el panel EC2. Seleccionamos la instancia del bastión, pulsamos el botón ‘Conectar’.



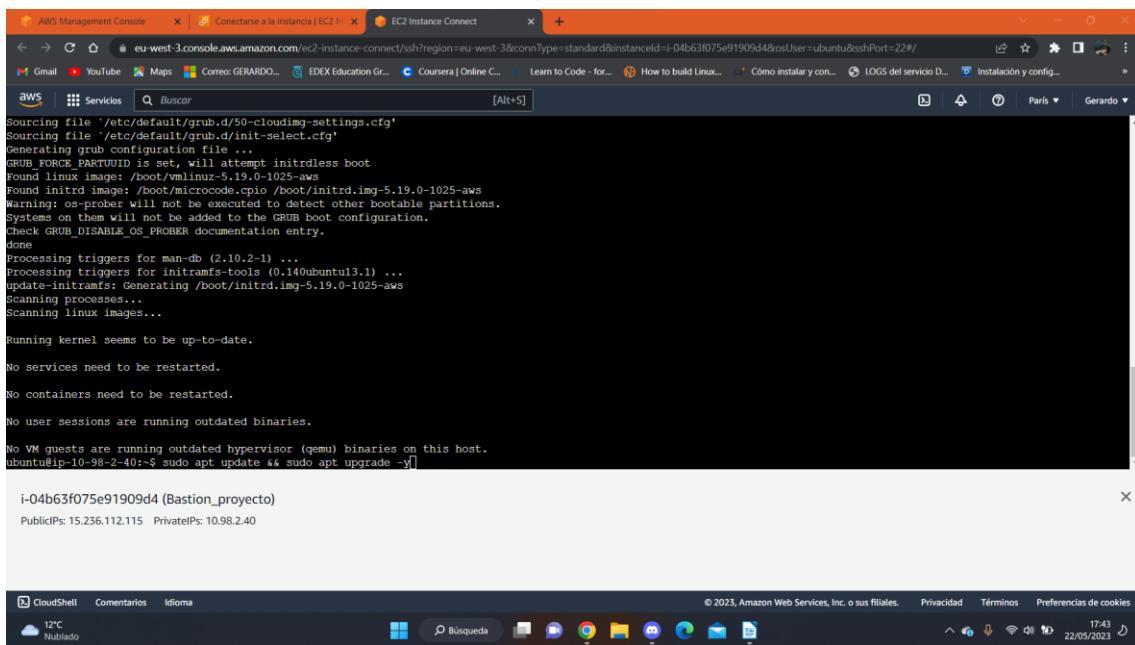
## Dockerización de un servidor completo en Cloud.

En la siguiente pantalla se nos muestran los detalles de la instancia, así como su nombre de usuario. Simplemente, hacemos click en ‘Conectar’.



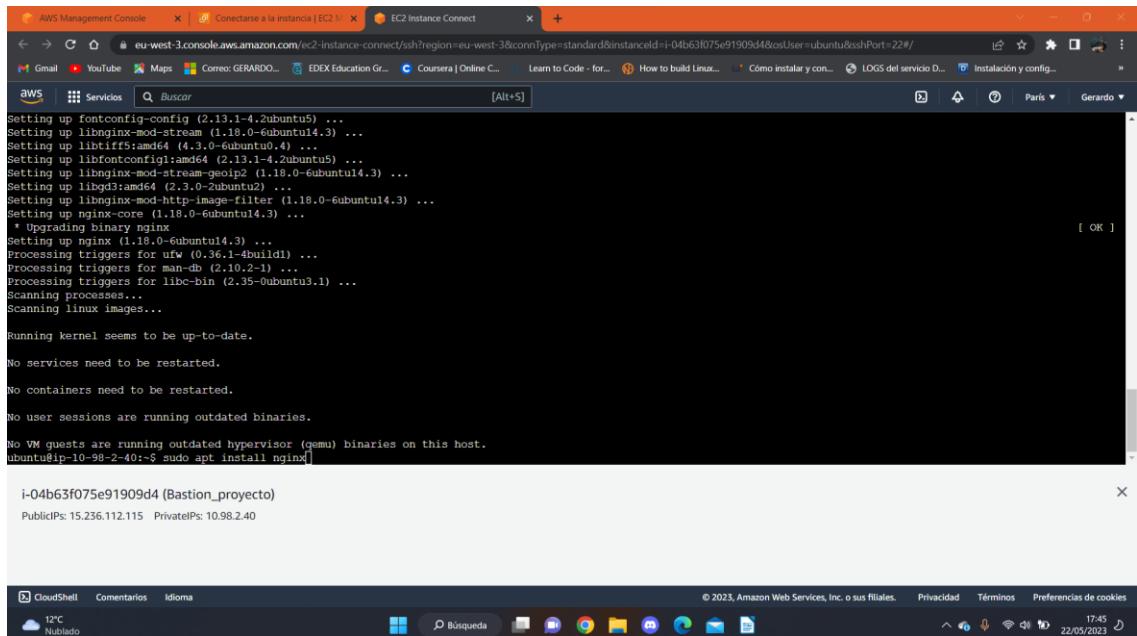
Una vez operativa la consola de la instancia, empezamos por actualizar el sistema. Como en cualquier sistema Ubuntu, ejecutaremos los clásicos comandos de actualización.

→ *sudo apt update && sudo apt upgrade -y*



Una vez finalizada la actualización, procedemos a instalar ‘Nginx’.

→ *sudo apt install nginx*



```
Setting up fontconfig-config (2.13.1-4.2ubuntu5) ...
Setting up libnginx-mod-stream (1.18.0-6ubuntu14.3) ...
Setting up libtiff5:amd64 (4.3.0-6ubuntu0.4) ...
Setting up libfontconfig1:amd64 (2.13.1-4.2ubuntu5) ...
Setting up libnginx-mod-stream-geoip2 (1.18.0-6ubuntu14.3) ...
Setting up libqd3:amd64 (2.3.0-2ubuntu2) ...
Setting up libnginx-mod-http-image-filter (1.18.0-6ubuntu14.3) ...
Setting up nginx-core (1.18.0-6ubuntu14.3) ...
 * Upgrading binary nginx
Setting up nginx (1.18.0-6ubuntu14.3) ...
Processing triggers for ufw (0.36.1-4build1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.

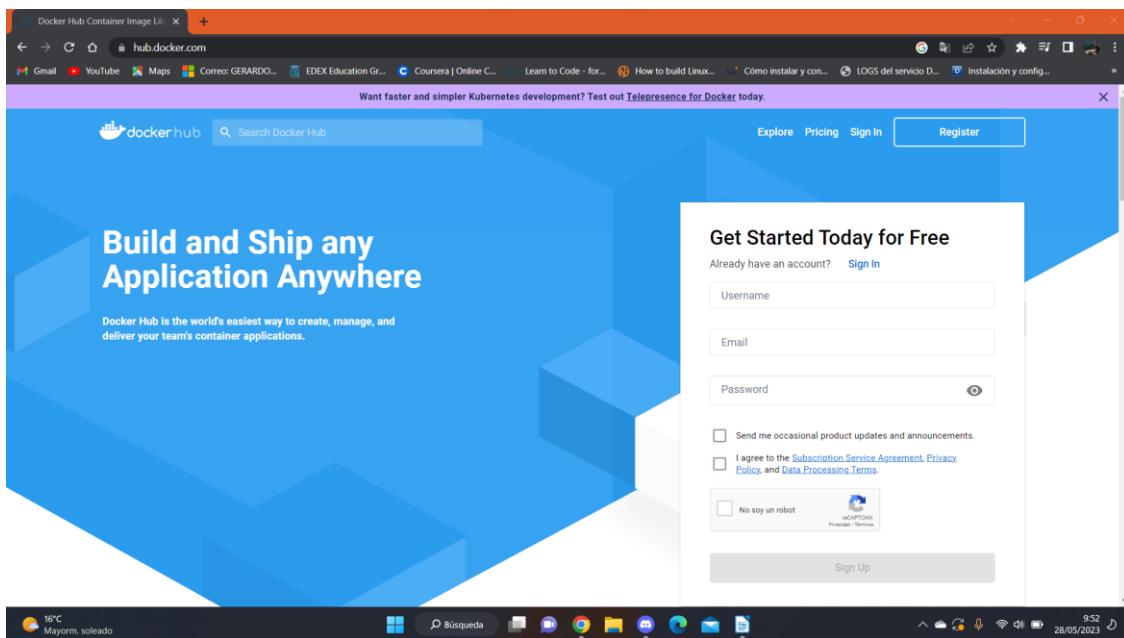
ubuntu@ip-10-98-2-40:~$ sudo apt install nginx[
```

Concluida la instalación, finaliza este manual. La configuración de Nginx como proxy inverso se realizará más adelante junto a la configuración de cada servicio del servidor. Pulsa [aquí](#) para volver arriba.

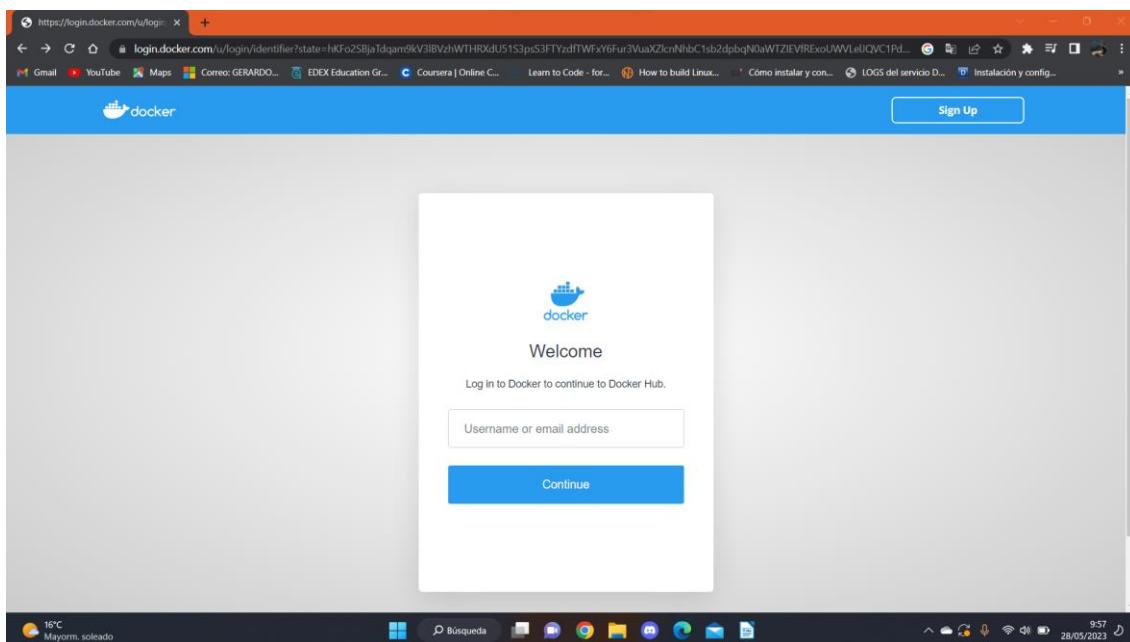
## 10.6 Anexo 6: Manual de registro en Docker Hub.

Para crear una cuenta en Docker Hub y crear un repositorio donde subir imágenes Docker seguiremos los siguientes pasos:

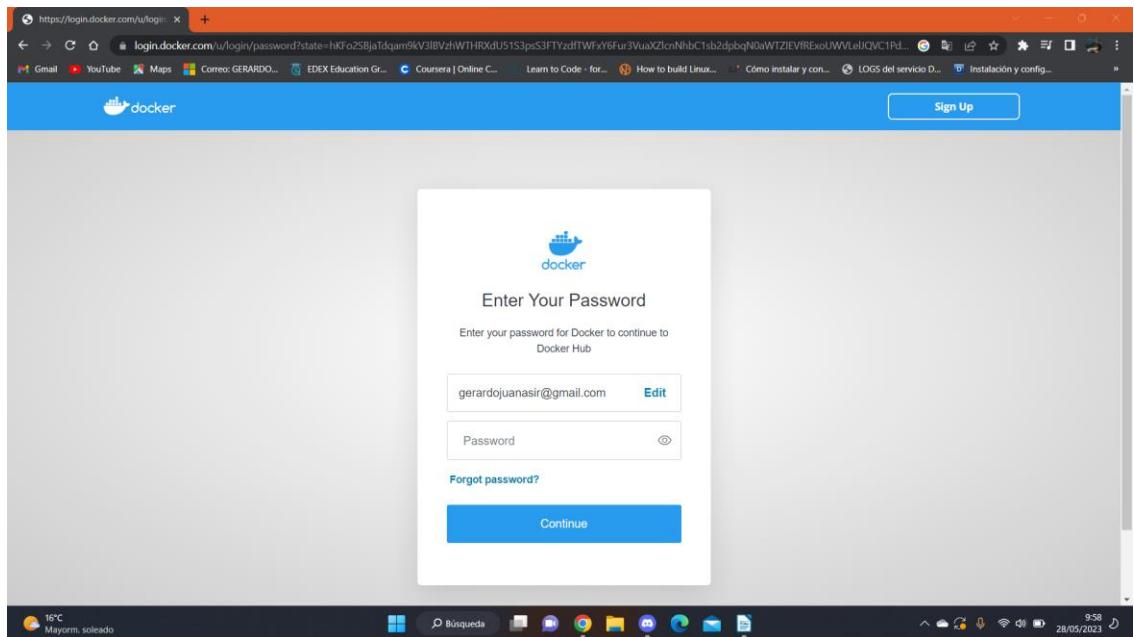
1. Desde un navegador accedemos a '<https://hub.docker.com>'. Una vez en la página, introducimos nuestro nombre de usuario, mail y contraseña. Aceptamos los términos de servicio y resolvemos el captcha. Finalmente, clickamos en 'Sign Up'.



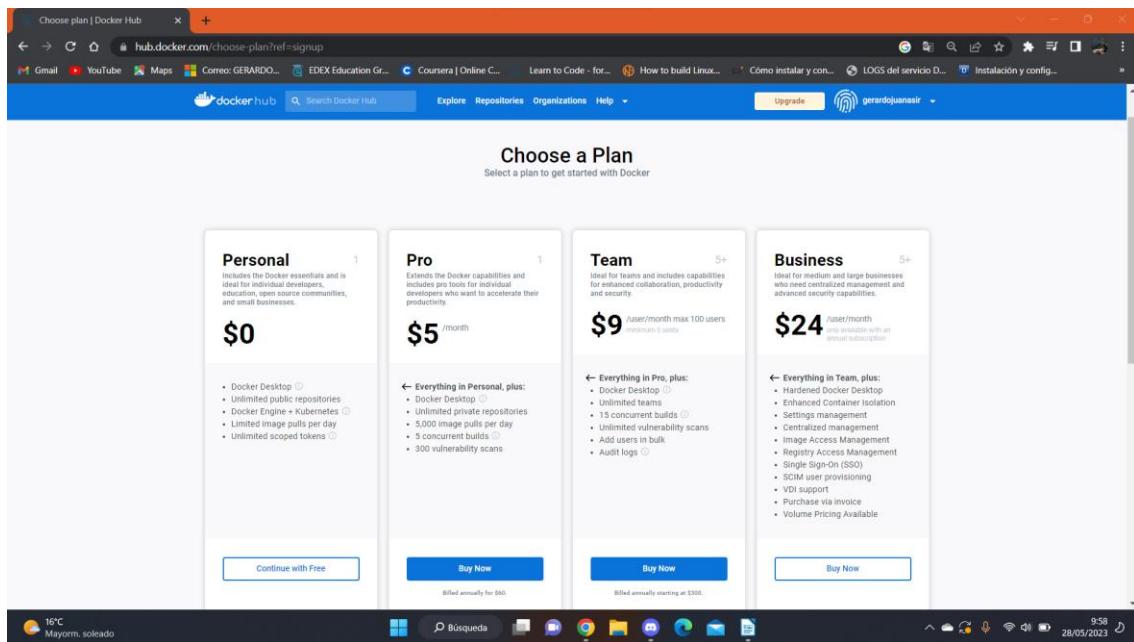
2. En la siguiente ventana se nos pedirá que introduzcamos el mail que acabamos de introducir.



3. Así, en esta nueva pantalla sólo tendremos que escribir la contraseña.

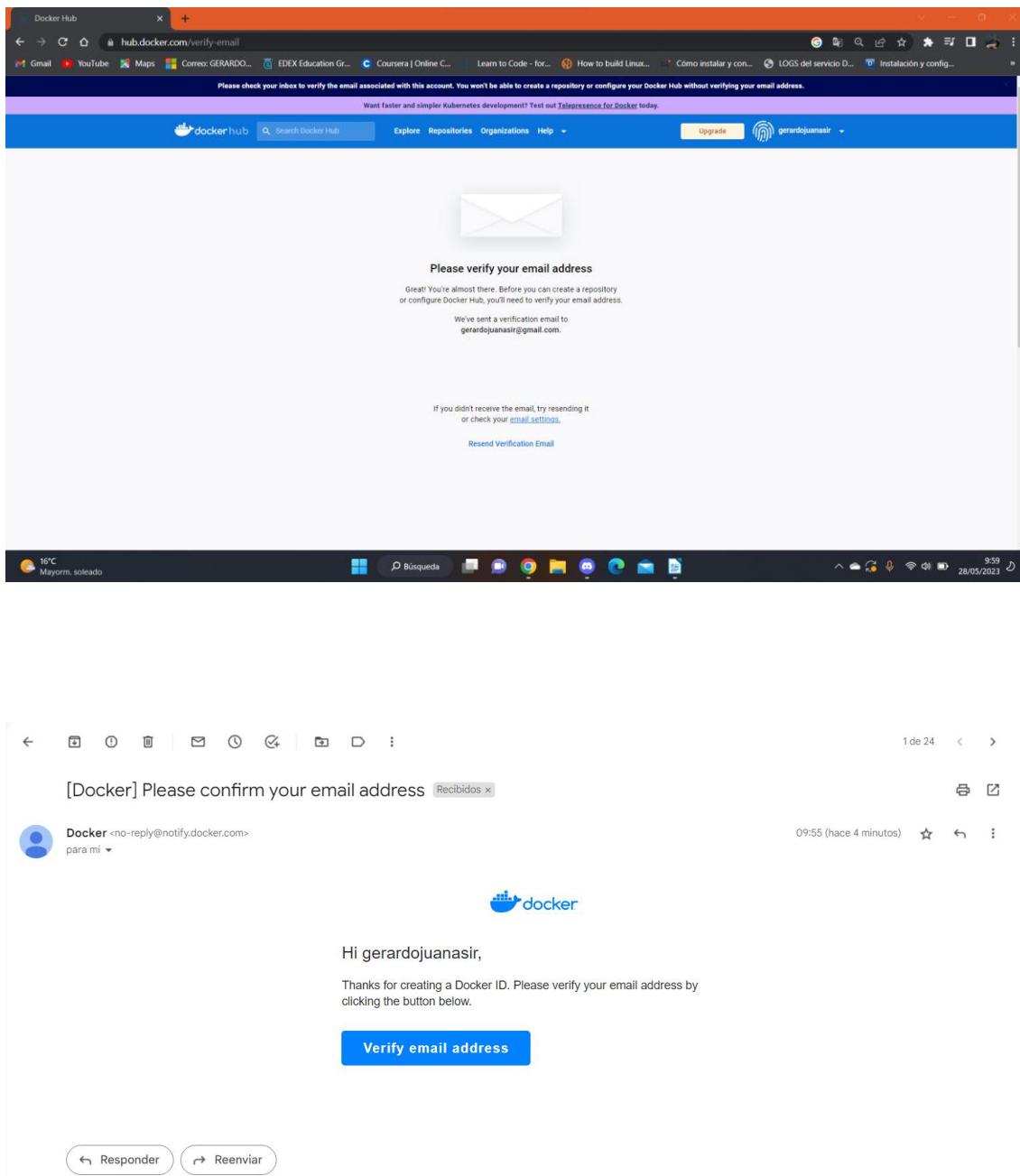


4. Docker Hub nos pedirá ahora elegir un plan de servicio. Como en este proyecto estamos usando lo máximo posible las capas gratuitas, elegimos el plan 'Personal'. Por lo tanto, hacemos click en 'Continue with Free'.

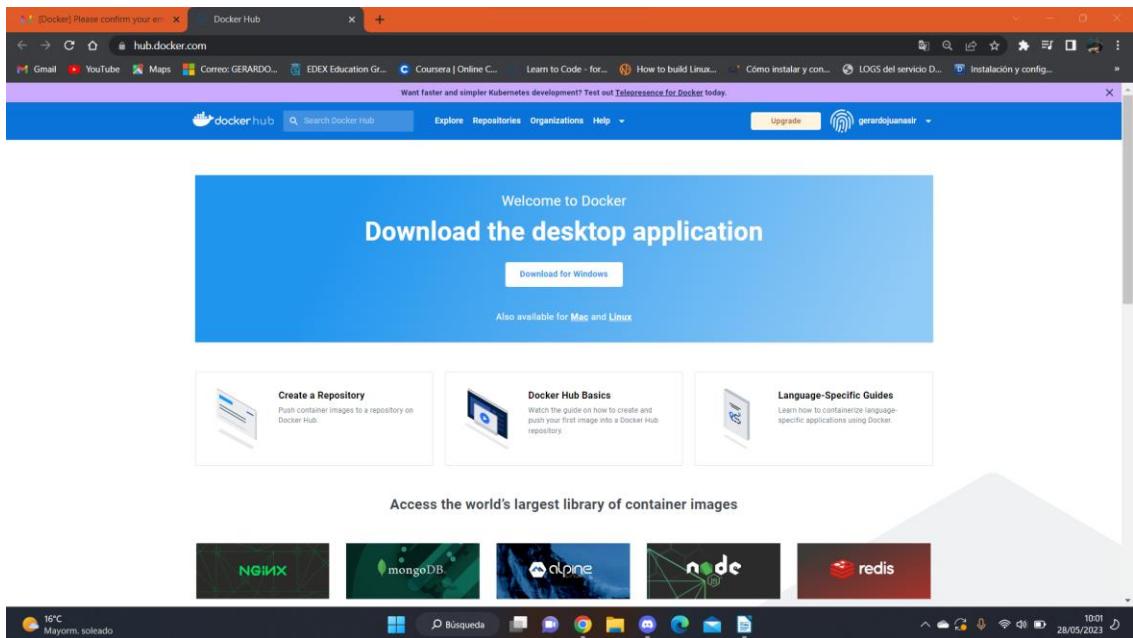


## Dockerización de un servidor completo en Cloud.

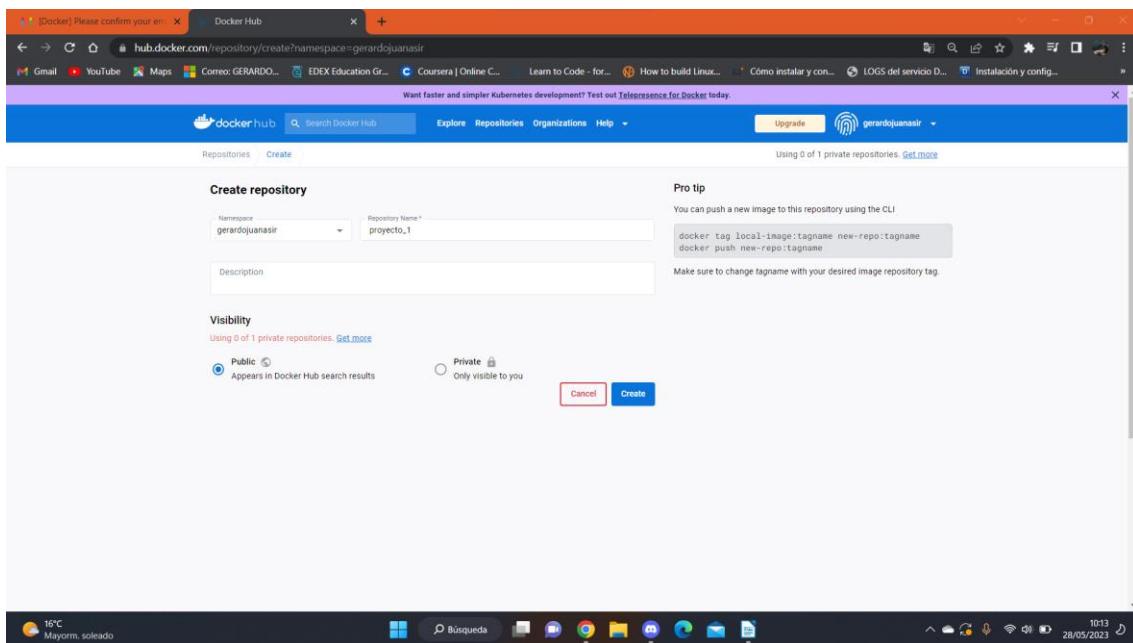
5. Se nos pedirá ahora que verifiquemos el mail. Para ello, tendremos que acudir a nuestra cuenta de correo, abrir el mail que nos ha llegado de Docker Hub y pulsar en ‘Verify email address’.



6. De esta forma ya tendremos operativa la cuenta en Docker Hub. Ahora vamos a crear un repositorio. Para ello, en la pantalla a la que se nos redirige al verificar el mail, pulsamos en ‘Create a Repository’.

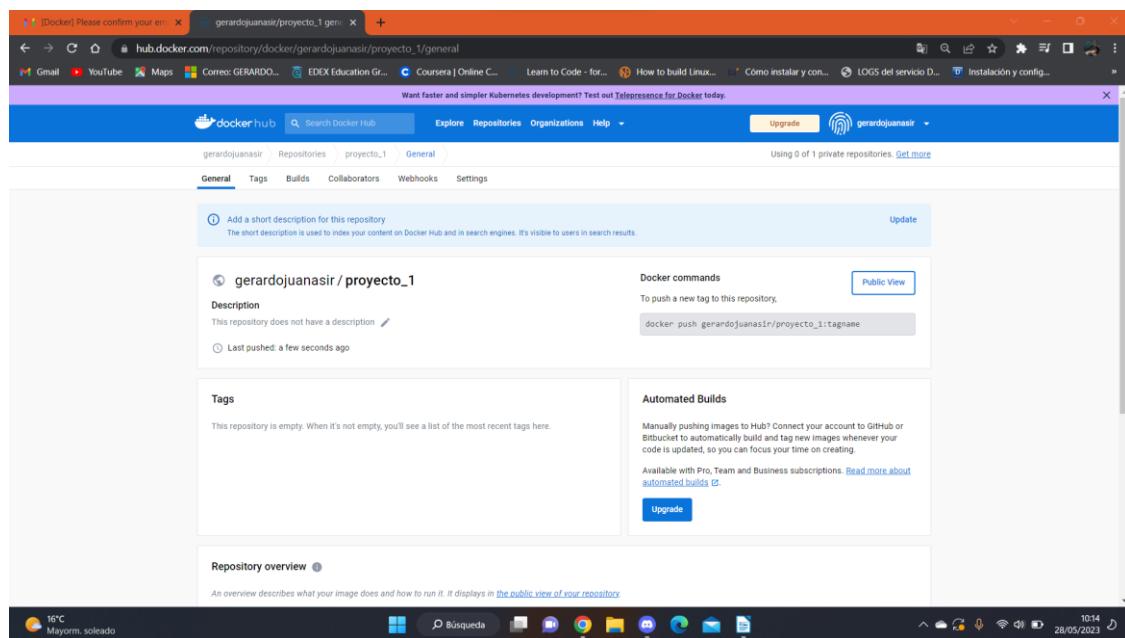


7. Seleccionamos en ‘Namespace’ la cuenta a la que pertenecerá el repositorio. A continuación, introducimos un nombre para el repositorio. Podemos agregarle una descripción, aunque esto es opcional. Finalmente, elegimos si queremos que nuestro repositorio sea público o privado y pulsamos en ‘Create’.



## Dockerización de un servidor completo en Cloud.

8. Con esto queda ya configurada la cuenta y el repositorio, listas para la subida de imágenes Docker que tendrá lugar más tarde.



Pulsa [aquí](#) para volver arriba.

## 10.7 Anexo 7: Manual de instalación de Docker en Ubuntu 22.04.

Estos son los pasos a seguir para instalar Docker tanto en la instancia-servidor como en Ubuntu 22.04 virtualizado.

1. Empezamos por actualizar los repositorios del sistema. Después, instalamos una serie de paquetes relacionados con la propia gestión de paquetes y la seguridad en la descarga de software.

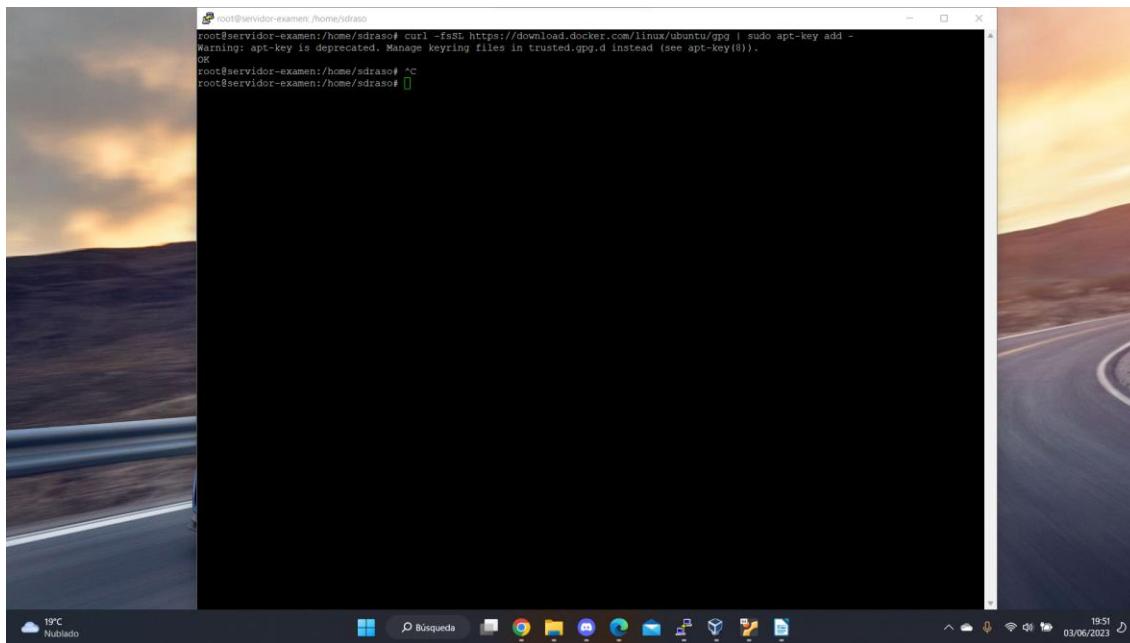
```
→ sudo apt update
→ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

```
root@servidor-examen:~# sudo apt update
Leyendo lista de paquetes... Hecho
Creando arbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20230311ubuntu0.22.04.1).
fijado ca-certificates como instalado manualmente.
curl ya está en su versión más reciente (7.81.0-1ubuntu1.10).
fijado curl como instalado manualmente.
software-properties-common ya está en su versión más reciente (0.99.22.7).
fijado software-properties-common como instalado manualmente.
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
libflashrom1 libftd1-2
Utilice sudo apt autoremove para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
curl 7.81.0-1ubuntu1.10
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 1.510 B de archivos.
Se utilizarán 169 KB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des: http://es.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.9 [1.510 B]
Descargando apt-transport-https (2.4.9) ...
Seleccionando el paquete apt-transport-https previamente no seleccionado.
(Leyendo la base de datos ... 111576 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../apt-transport-https_2.4.9_all.deb ...
Desempaquetando apt-transport-https (2.4.9) ...
Configurando apt-transport-https (2.4.9) ...
Scanning processes...
Scanning candidates...
Scanning linux Images...
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@servidor-examen:~#
```

## Dockerización de un servidor completo en Cloud.

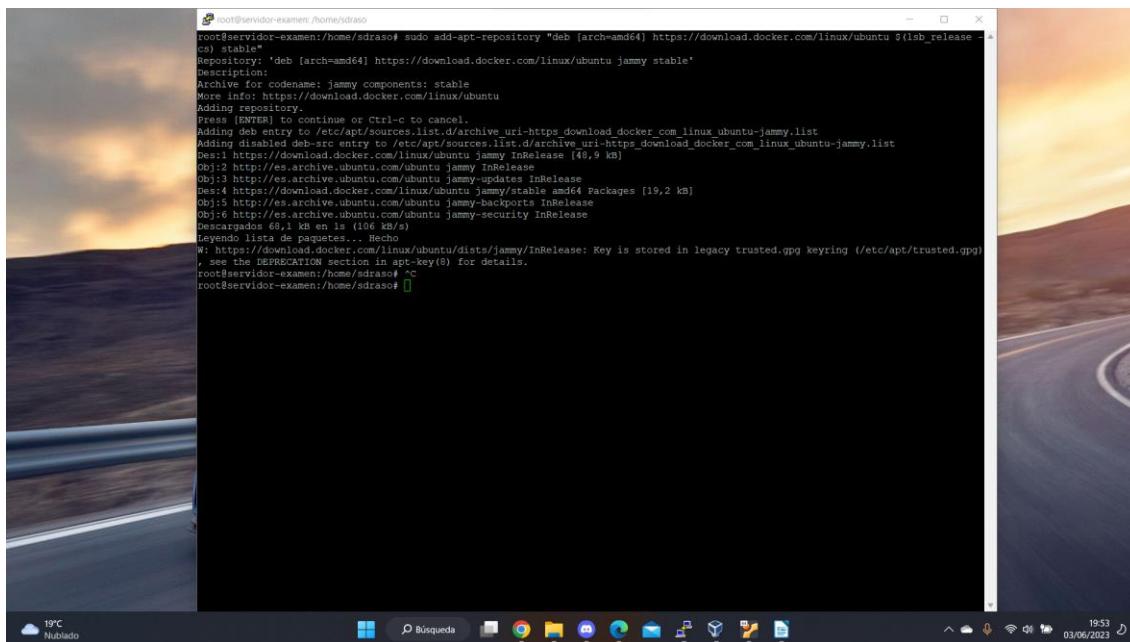
2. A continuación, se descarga el archivo GPG de Docker y se agrega al sistema de claves APT de la máquina.

→ `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`



3. Ahora se agrega el repositorio de Docker al sistema de gestión de paquetes APT de Ubuntu:

→ `sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu ${lsb_release -cs} stable"`



4. Volvemos a actualizar e instalamos Docker:

- `sudo apt update`
- `sudo apt-install docker-ce`

```

root@servidor-examen:/home/sdrasco
[sudo] password for root:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  docker-ce docker-ce-rootless-extras docker-ce-cli docker-ce-cli-plugin
  docker-compose-plugin libslirp0 libslirp0-amd64
0 upgraded, 6 new, 0 to remove and 0 not upgraded.
Need to get 1,032 B/1,032 B from 1 source.
Get:1 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce
[1032/1032 B]
Preconfiguring packages ...
Selecting previously unselected package docker-ce-buildx-plugin.
Preparing to unpack .../docker-ce-buildx-plugin_0.10.5-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-buildx-plugin.
Preparing to unpack .../2-docker-buildx-plugin_0.10.5-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-ce-clion.
Preparing to unpack .../3-docker-ce-clion_513a24.0.2-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../4-docker-ce-rootless-extras_513a24.0.2-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../5-docker-ce_513a24.0.2-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../5-docker-ce-rootless-extras_513a24.0.2-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-ce-clion.
Preparing to unpack .../6-docker-ce-clion_513a24.0.2-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package docker-compose-plugin.
Preparing to unpack .../7-docker-compose-plugin_2.18.1-1ubuntu.22.04-jammy_amd64.deb ...
Selecting previously unselected package libslirp0.
Preparing to unpack .../8-libslirp0_4.6.1-1build1_amd64.deb ...
Selecting previously unselected package libslirp0-amd64.
Preparing to unpack .../9-libslirp0-amd64_4.6.1-1build1_amd64.deb ...
Selecting previously unselected package libslirp0-amd64.
Preparing to unpack .../10-libslirp0-amd64_4.6.1-1build1_amd64.deb ...
Configuring docker-buildx-plugin (0.10.5-1ubuntu.22.04-jammy) ...
Configuring containerd.io (1.6.21-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Configuring docker-ce (5:24.0.2-1ubuntu.22.04-jammy) ...
Configuring docker-ce-clion (5:24.0.2-1ubuntu.22.04-jammy) ...
Configuring docker-ce-rootless-extras (5:24.0.2-1ubuntu.22.04-jammy) ...
Configuring docker-ce (5:24.0.2-1ubuntu.22.04-jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Procesando disparadores para libc-bin (2.35-0ubuntu3.1) ...
Procesando disparadores para libslirp0 (2.35-0ubuntu3.1) ...
Scanning processes...
Scanning containers...
Scanning linux images...
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (gemm) binaries on this host.
W: Se interrumpió la operación antes de que pudiera terminar
root@servidor-examen:/home/sdrasco# sudo apt install docker-ce

```

5. Comprobamos que Docker está instalado el sistema consultando su versión.

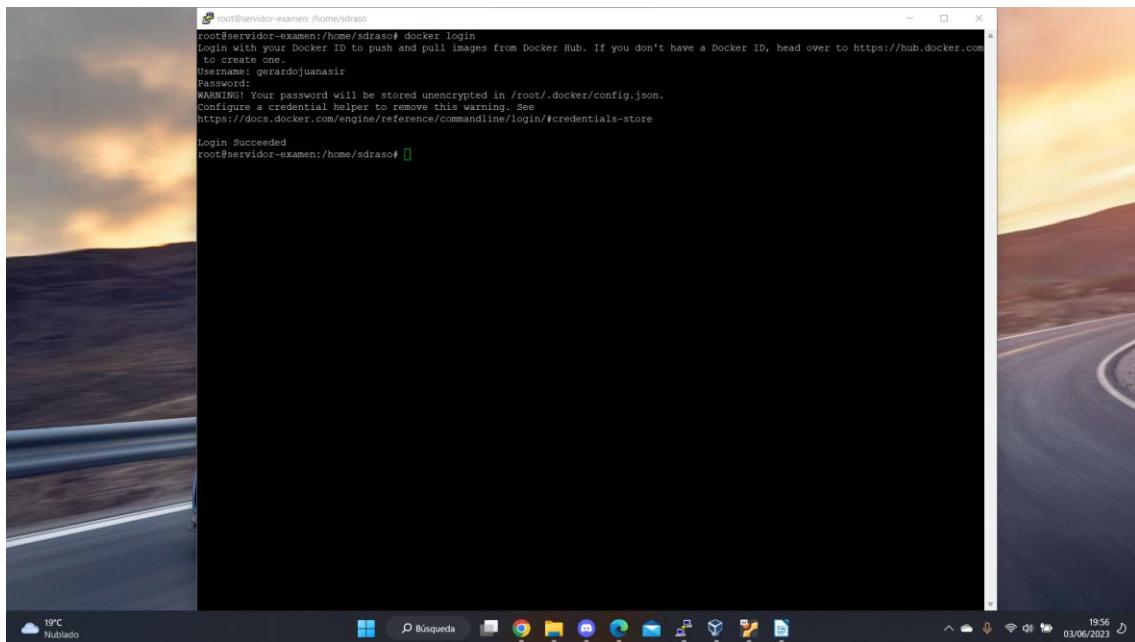
```

root@servidor-examen:/home/sdrasco
root@servidor-examen:/home/sdrasco# docker --version
Docker version 24.0.2, build cb7edfc
root@servidor-examen:/home/sdrasco#

```

6. Finalmente, para que más tarde podamos subir las imágenes Docker al repositorio que creamos previamente en Docker Hub, tenemos que hacer login desde el sistema. Se nos preguntará el nombre de usuario y la contraseña.

→ `sudo su`  
→ `docker login`

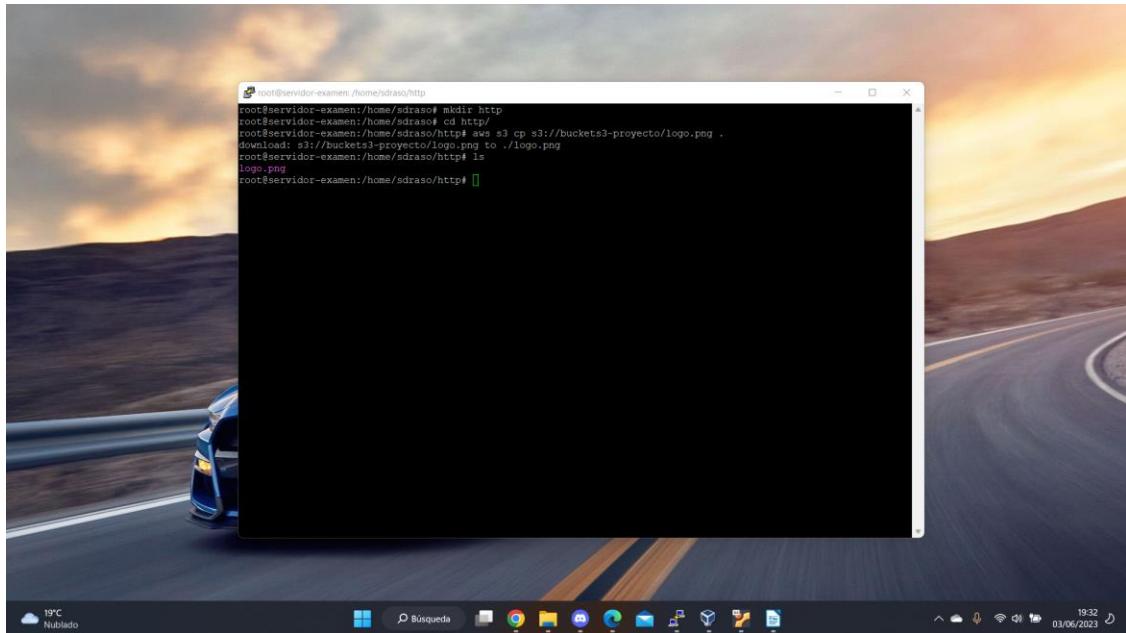


Con esto finaliza el manual. Pulsa [aquí](#) para volver arriba.

## 10.8 Anexo 8: Manual HTTP.

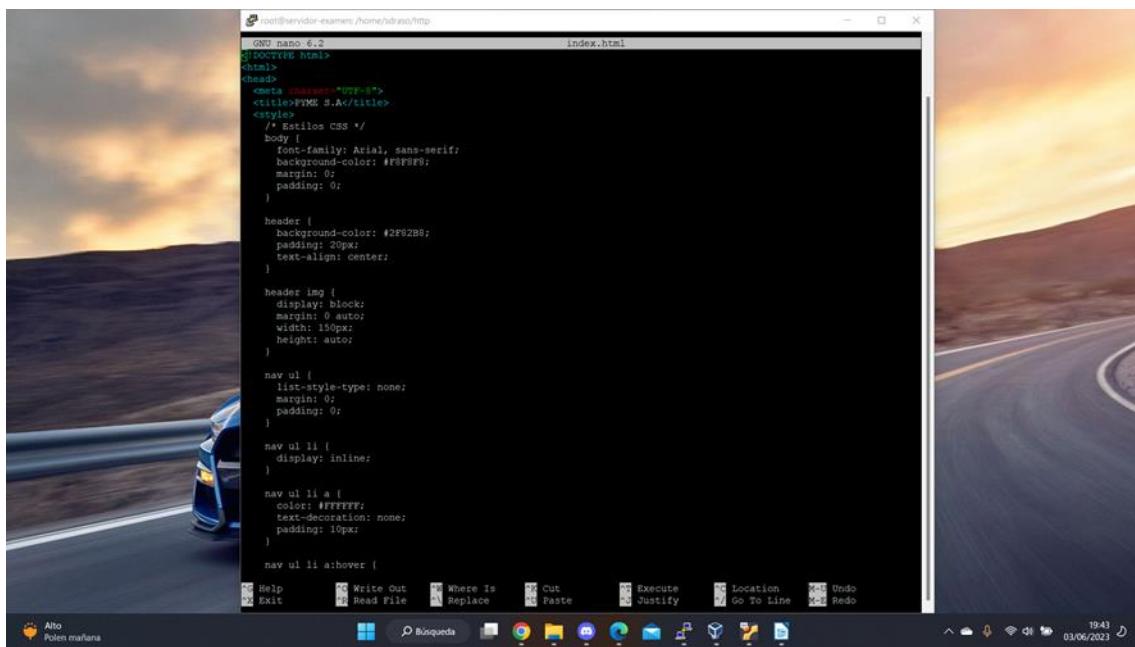
El servicio HTTP no requiere la previa obtención de ningún fichero de configuración, ya que usaremos la instalación base de apache2 en la imagen. Empezamos creando una carpeta para el trabajo que engloba este manual. Después, nos descargamos del bucket el logo de la empresa, necesario para el diseño de la página web.

```
→ mkdir http  
→ cd http  
→ aws s3 ls s3://buckets3-proyecto  
→ aws s3 cp s3://buckets3-proyecto/logo.png
```



## Dockerización de un servidor completo en Cloud.

Una vez tengamos el logo en la carpeta, creamos el archivo ‘index.html’, que contiene el código de la web que mostrará apache2.



```
root@servidor-examen:/home/distro/http
GNU nano 6.2                               index.html
[1] 2024TUE 09:43
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>PYME S.A.</title>
<style>
/* Estilos CSS */
body {
    font-family: Arial, sans-serif;
    background-color: #F8F8F8;
    margin: 0;
    padding: 0;
}

header {
    background-color: #2F82B8;
    padding: 20px;
    text-align: center;
}

header img {
    display: block;
    margin: 0 auto;
    width: 150px;
    height: auto;
}

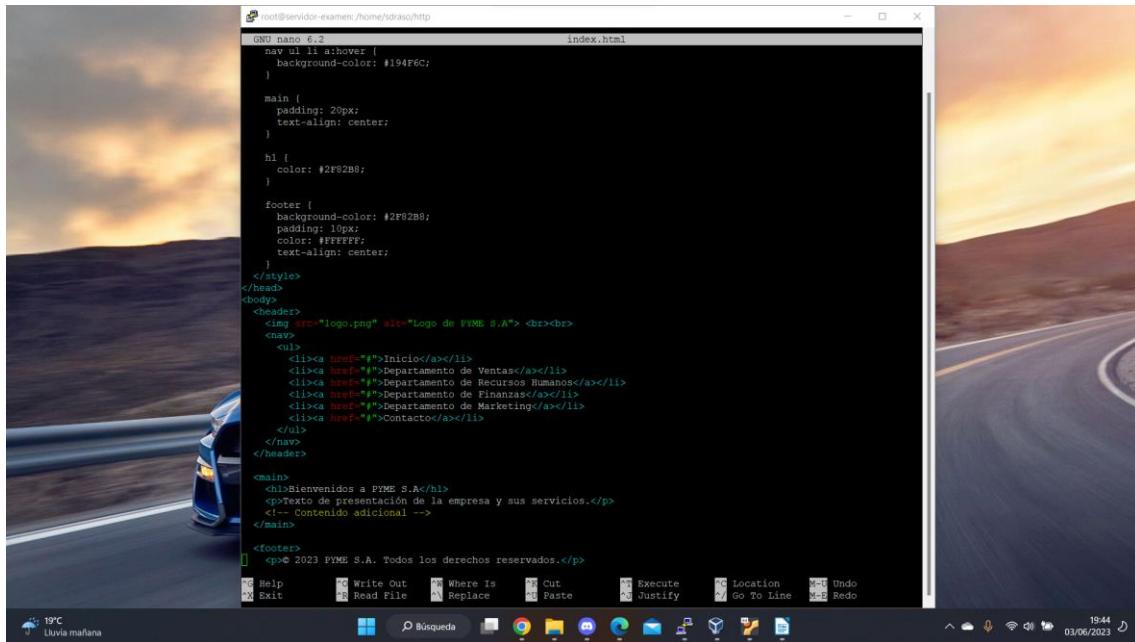
nav ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

nav ul li {
    display: inline;
}

nav ul li a {
    color: #FFFFFF;
    text-decoration: none;
    padding: 10px;
}

nav ul li a:hover {
    background-color: #194F6C;
}

```



```
root@servidor-examen:/home/distro/http
GNU nano 6.2                               index.html
[1] 2024TUE 09:43
nav ul li a:hover {
    background-color: #194F6C;
}

main {
    padding: 20px;
    text-align: center;
}

h1 {
    color: #2F82B8;
}

footer {
    background-color: #2F82B8;
    padding: 10px;
    color: #FFFFFF;
    text-align: center;
}

```

```
</style>
</head>
<body>
<header>
     <br><br>
    <nav>
        <ul>
            <li><a href="#">Inicio</a></li>
            <li><a href="#">Departamento de Ventas</a></li>
            <li><a href="#">Departamento de Recursos Humanos</a></li>
            <li><a href="#">Departamento de Finanzas</a></li>
            <li><a href="#">Departamento de Marketing</a></li>
            <li><a href="#">Contacto</a></li>
        </ul>
    </nav>
</header>

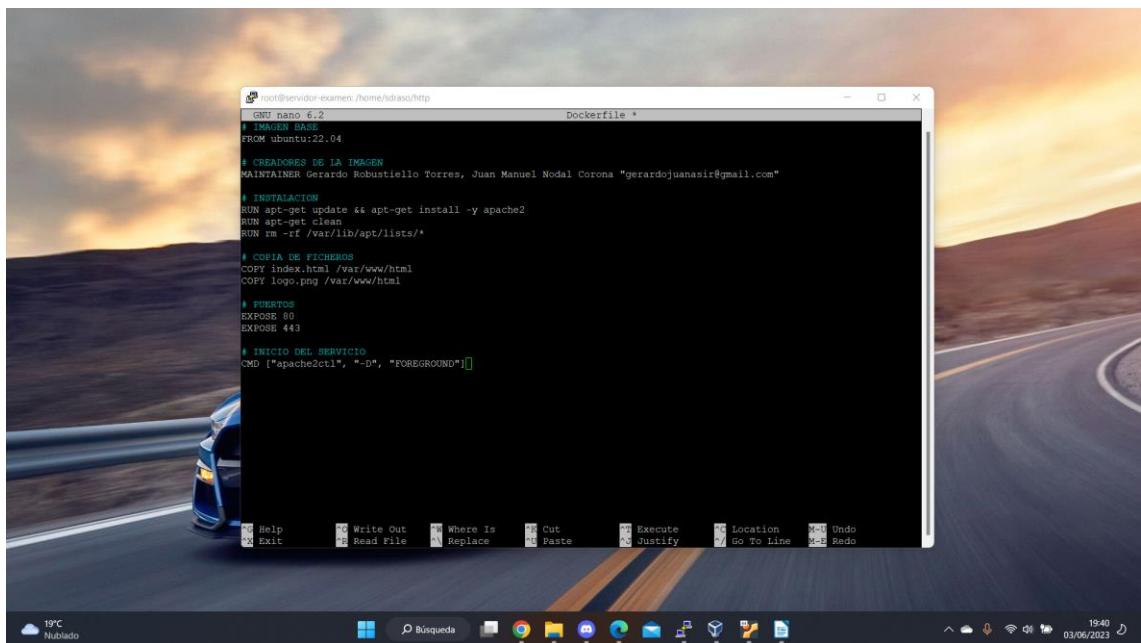
<main>
    <h1>Bienvenidos a PYME S.A.</h1>
    <p>Texto de presentación de la empresa y sus servicios.</p>
    <!-- Contenido adicional -->
</main>

<footer>
    <p>© 2023 PYME S.A. Todos los derechos reservados.</p>

```

```
<footer>
    <p>© 2023 PYME S.A. Todos los derechos reservados.</p>
</footer>
</body>
</html>
```

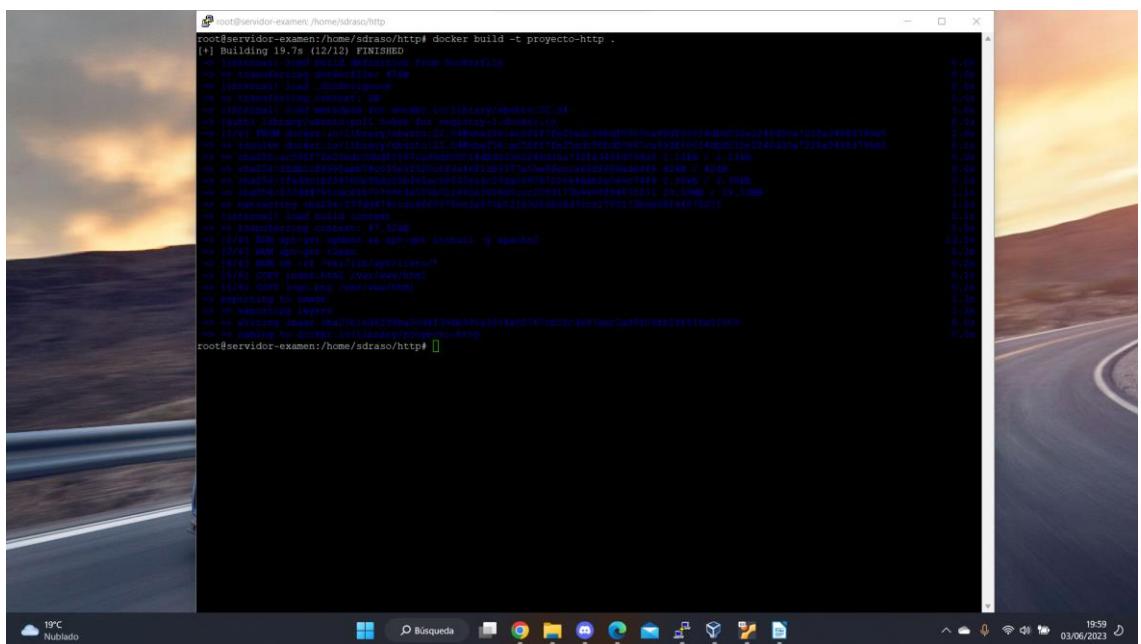
Ahora vamos a escribir el Dockerfile, que es el fichero que contiene las instrucciones de creación de la imagen. Importante recordar que el fichero no tiene ninguna extensión.



Así, le estamos diciendo que construya una imagen basada en Ubuntu 22.04, le indicamos los creadores de la imagen, le decimos que actualice sus repositorios e instale apache2, borramos restos innecesarios de la instalación, copiamos a la imagen el index y el logo, abrimos los puertos 80 y 443 y, finalmente, indicamos el comando que ejecutará apache2 al iniciarse el contenedor.

Una vez tenemos el Dockerfile, procedemos a crear la imagen. Para ello utilizamos el siguiente comando:

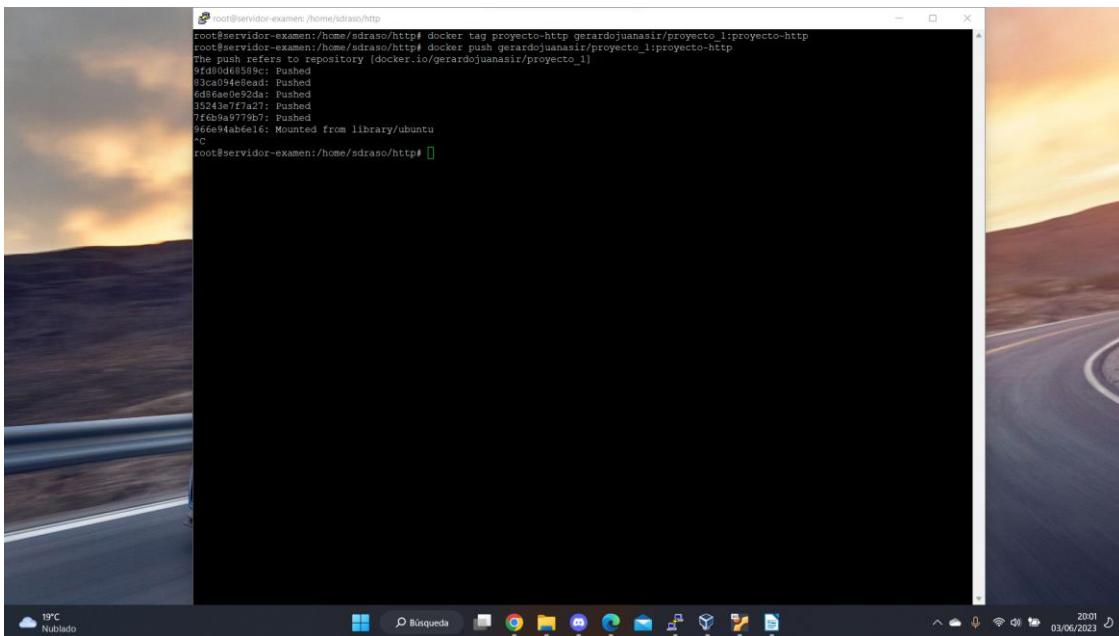
→  *docker build -t proyecto-http*



## Dockerización de un servidor completo en Cloud.

Lo siguiente es etiquetar la imagen para identificarla y subirla a nuestro repositorio en Docker Hub. Con tag se etiqueta y con push se sube al repositorio.

- `docker tag proyecto-http gerardojuanasir/proyecto_1:proyecto-http`
- `docker push gerardojuanasir/proyecto_1:proyecto-http`



Pasamos al bastión. Para que el servicio HTTP de nuestro futuro servidor funcione correctamente, debemos configurar las reglas del proxy inverso que permitan el reenvío de las peticiones de los clientes. En la siguiente captura se ha accedido al bastión por conexión SSH con el siguiente comando:

- `ssh -i "clave_bastion.pem" ubuntu@15.236.112.115`

Donde 'clave\_bastion.pem' es el fichero de credenciales, Ubuntu es el nombre de usuario del bastión y la IP es la IP elástica del bastión

Abrimos el fichero de configuración de Nginx y añadimos las líneas incluidas en la directiva server.

Linux\_Mint\_Cliente\_proyecto [Corriendo] - Oracle VM VirtualBox

Archivo Maquina Ver Entrada Dispositivos Ayuda

GNU nano 6.2 root@ip-10-98-2-40: /home/ubuntu

```
# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
# gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

## Virtual Host Configs
##

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;

server {
    listen 80;
    server_name 15.236.112.115;

    location / {
        proxy_pass http://10.98.1.193:80;
    }
}

#mail {
#    # See sample authentication script at
#}
```

Help Write Out Where Is Cut Execute Location Undo Set Mark To Bracket

Exit Read File Replace Paste Justify Go To Line Redo Copy Where Was

19°C Nublado

Búsquedas

Windows Derecha

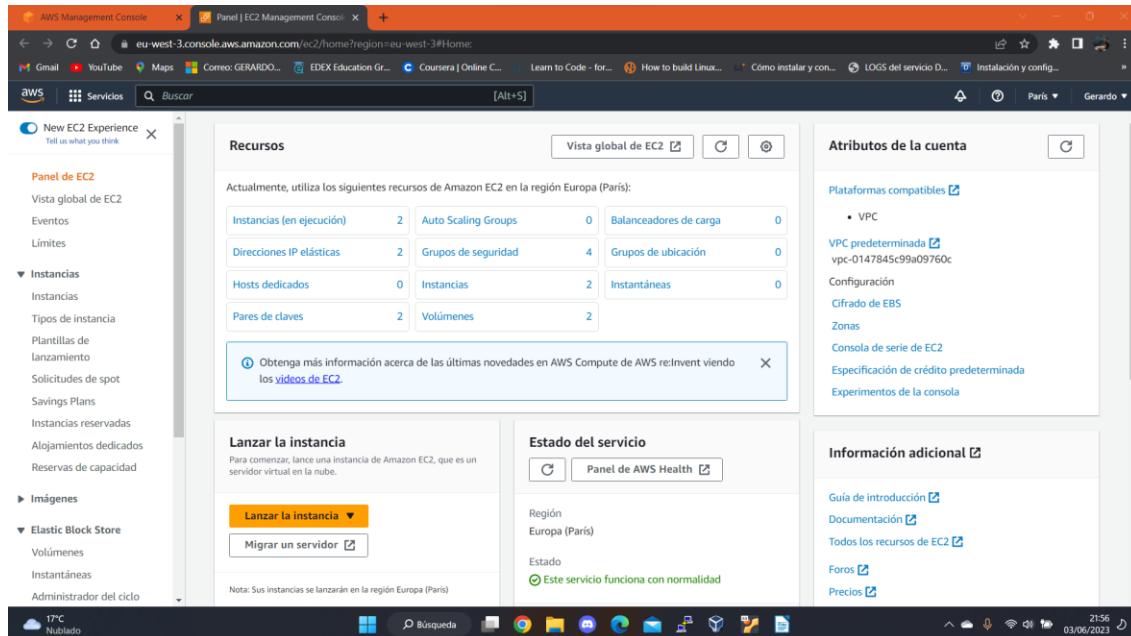
A continuación, reiniciamos el servicio de Nginx y comprobamos su estado por si hubiese errores.

```
→ sudo systemctl restart nginx  
→ sudo systemctl status nginx
```

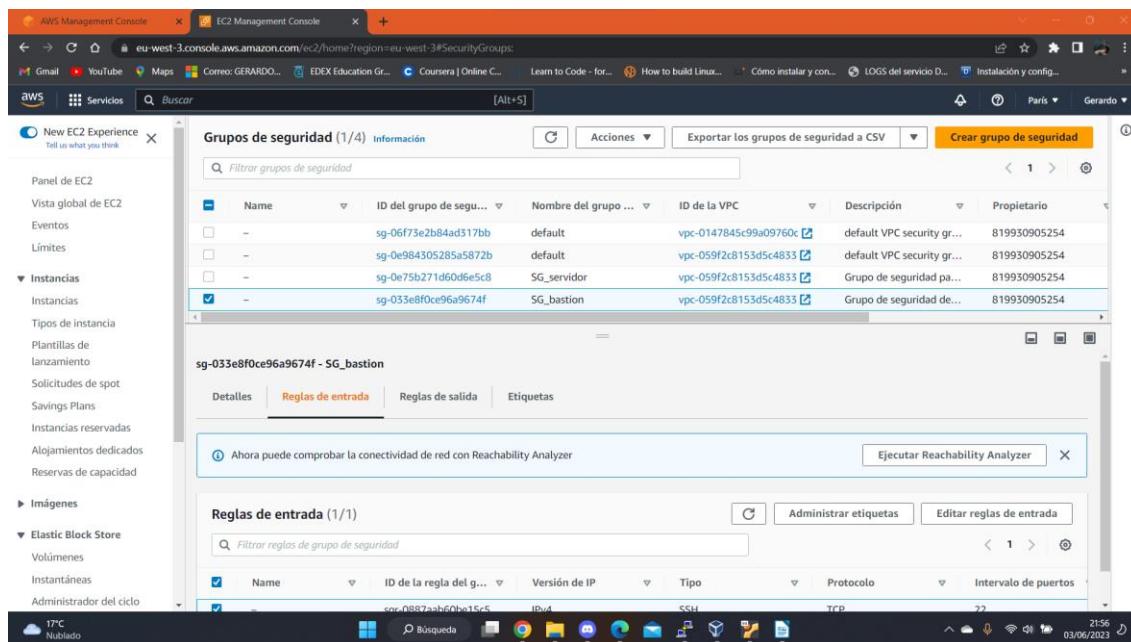
## Dockerización de un servidor completo en Cloud.

Por último, debemos configurar los grupos de seguridad asociados a las instancias. Estos grupos controlan los puertos de las instancias, teniendo que configurar reglas de entrada y salida para permitir el tráfico. Empezamos por el grupo de seguridad asociado al bastión.

Desde la consola de AWS, vamos al menú ‘EC2’. En él, pulsamos ‘Grupos de seguridad’.



Seleccionamos el grupo del bastión y hacemos click en ‘Editar reglas de entrada’.



La regla SSH que viene por defecto permite la conexión SSH al bastión desde cualquier lugar. Como se necesita la clave pem, no supone un riesgo de seguridad. Añadimos dos reglas más para los protocolos HTTP y HTTPS, respectivamente. En ‘Origen’ hemos especificado la IP externa de nuestro router, para que así sólo alguien conectado a nuestra red pueda enviar tráfico al bastión. En caso de aplicar al proyecto a una PYME real, se especificaría su IP externa. Guardamos las reglas y seguimos.

ID de la regla del grupo de seguridad	Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional
sgr-0887aab60be15c594	SSH	TCP	22	Person... ▾	Conexion_SSH
-	HTTP	TCP	80	Person... ▾	IP_publica_Gerardo
-	HTTPS	TCP	443	Person... ▾	IP_publica_Gerardo

En cuanto a las reglas de salida, ponemos una única regla que permita todo el tráfico a todas direcciones para un correcto funcionamiento de la conexión a Internet del bastión.

ID de la regla del grupo de seguridad	Tipo	Protocolo	Intervalo de puertos	Destino	Descripción: opcional
sgr-009d43de88e92cac1	Todo el tráfico	Todo	Todo	Person... ▾	0.0.0.0/0

## Dockerización de un servidor completo en Cloud.

Pasamos al grupo de seguridad del servidor. En las reglas de entradas repetimos el mismo procedimiento, indicando en el ‘Origen’ la IP privada del bastión, para que solamente le entre tráfico desde él. Así, la seguridad del servidor se incrementa muchísimo.

ID de la regla del grupo de seguridad	Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional
sgr-03fc2a16e672c416d	SSH	TCP	22	Person... 10.98.2.40/32	SSH_bastion
-	HTTP	TCP	80	Person... 10.98.2.40/32	HTTP_bastion
-	HTTPS	TCP	443	Person... 10.98.2.40/32	HTTPS_bastion

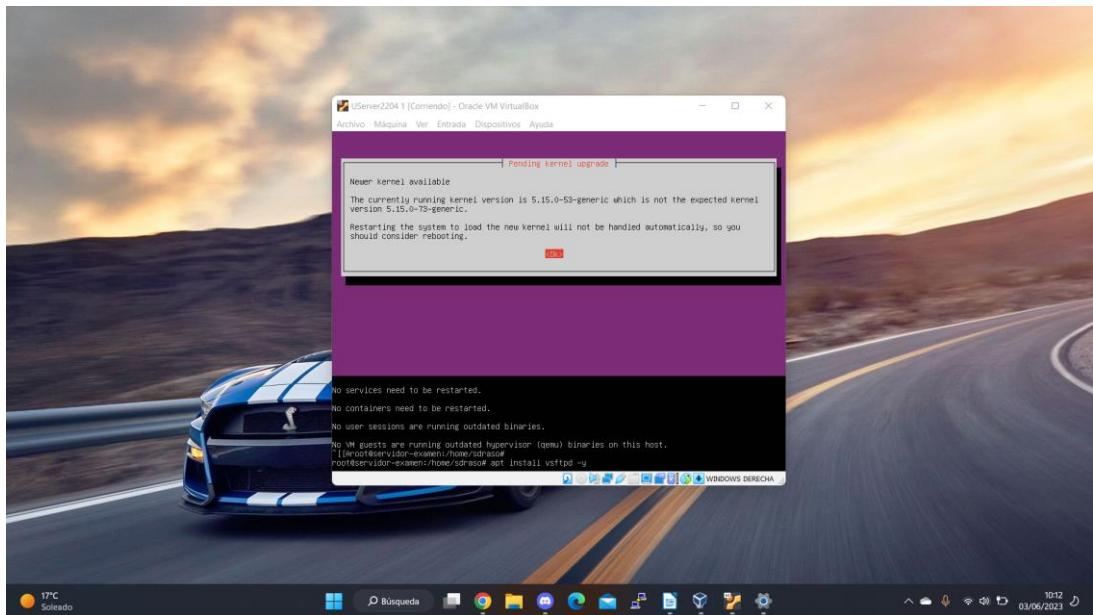
En cuanto a las reglas de salida, también permitimos todo el tráfico a todas partes.

Con esto queda lista la imagen y la infraestructura para la prueba final de la imagen HTTP. Pulsa [aquí](#) para volver arriba.

## **10.9 Anexo 9: Manual FTP.**

En un Ubuntu 22.04 virtualizado en red local, instalamos vsftpd para conseguir el fichero de configuración. Antes, actualizamos el servidor.

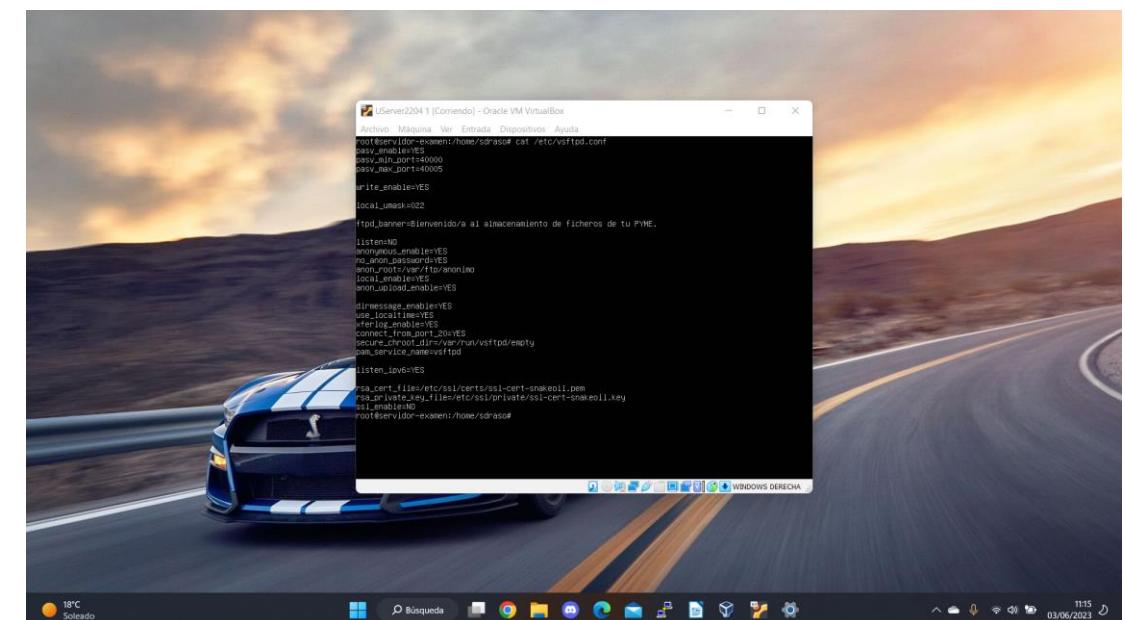
- ```
→ sudo su  
→ apt update && apt upgrade -y  
→ apt install vsftpd
```



Ahora editamos el fichero de configuración. Primero hacemos una copia de seguridad del mismo.

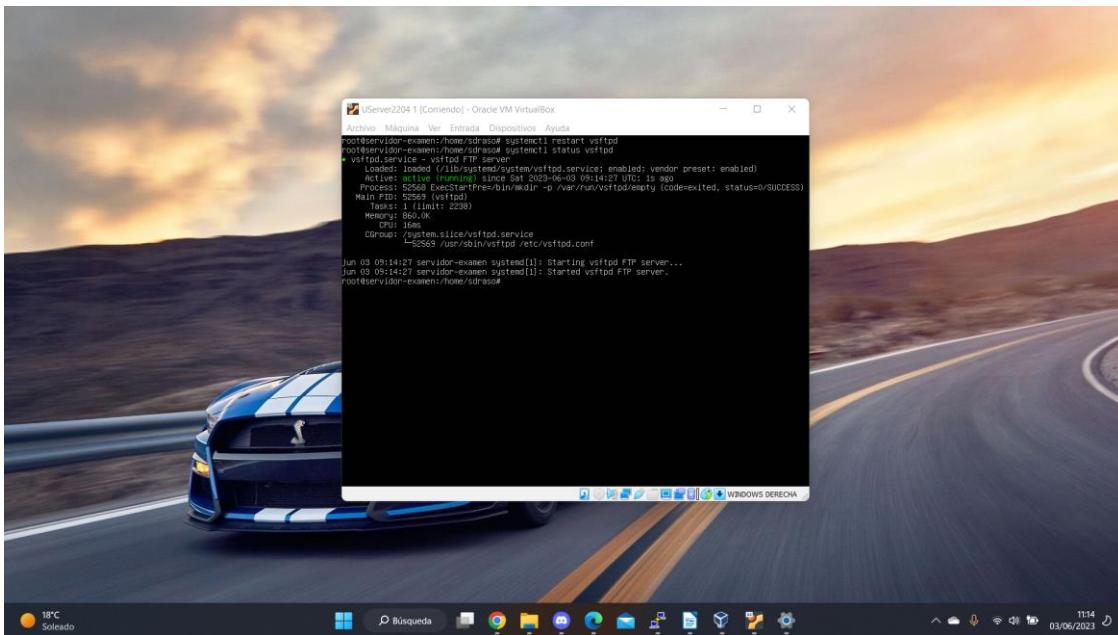
→ cp /etc/vsftpd.conf /etc/vsftpd.conf.default

En la configuración, habilitamos el modo pasivo y el modo anónimo. Agregamos también un mensaje de bienvenida.

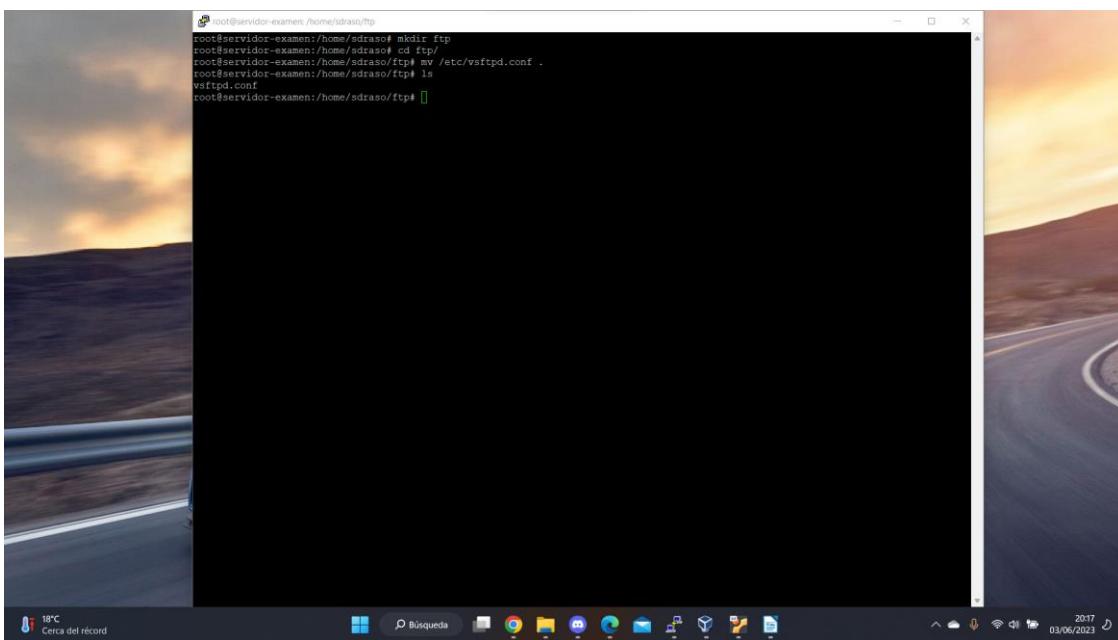


## Dockerización de un servidor completo en Cloud.

Probamos la configuración, comprobando que está en funcionamiento.



Ahora que ya tenemos el fichero de configuración es momento de pasar a la construcción de la imagen. Creamos una carpeta 'ftp' donde trabajar y movemos a ella el fichero de configuración de vsftpd.



A continuación, vamos a crear un script que permita el inicio del servicio cuando se ejecute la imagen en un contenedor. Lo llamaremos ‘entrypoint.sh’. Fijamos dentro un reinicio del servicio y añadimos el comando ‘tail -f /dev/null’, que mantendrá el contenedor en ejecución:

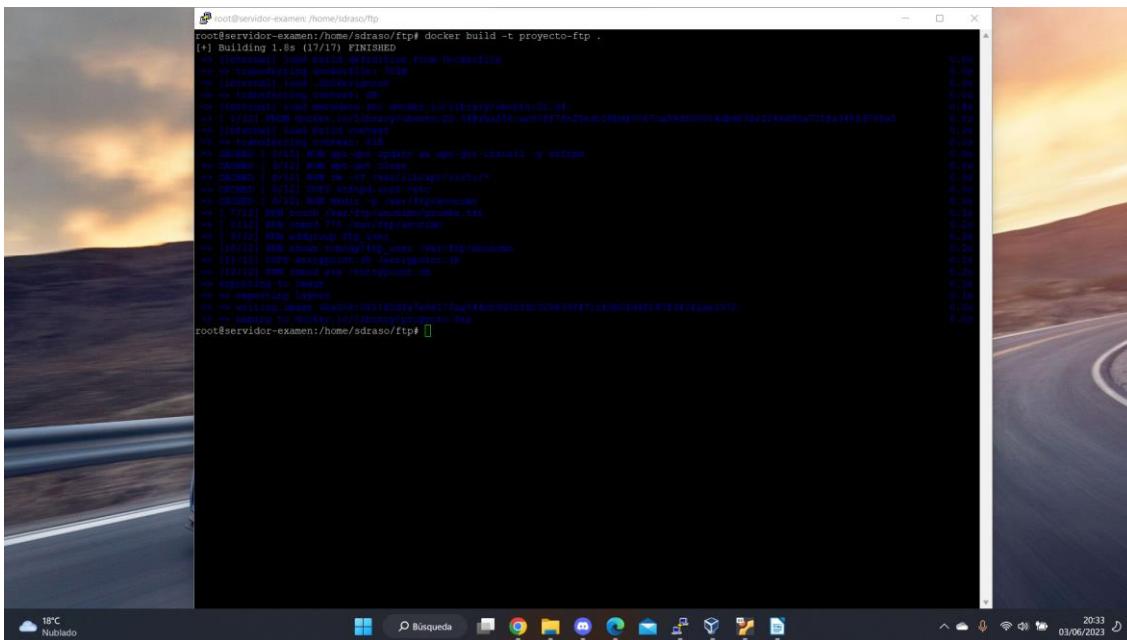
```
root@servidor-examen: /home/sdraso/ftp
GNU nano 6.2
#!/bin/bash
# INICIAR SERVICIO
service vsftpd restart
# MANTENER CONTENEDOR EJECUTANDOSE
tail -f /dev/null
```

Pasamos al Dockerfile. De nuevo tomamos una imagen Ubuntu 22.04 como base. Instalamos vsftpd, copiamos el fichero de configuración, creamos la ruta de donde se bajarán los ficheros los usuarios y modificamos sus permisos y dueños. Abrimos los puertos relacionados con el protocolo y copiamos el script de inicio de permisos, haciéndolo ejecutable.

```
FROM ubuntu:22.04
COPY vsftpd.conf /etc
COPY entrypoint.sh ./entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

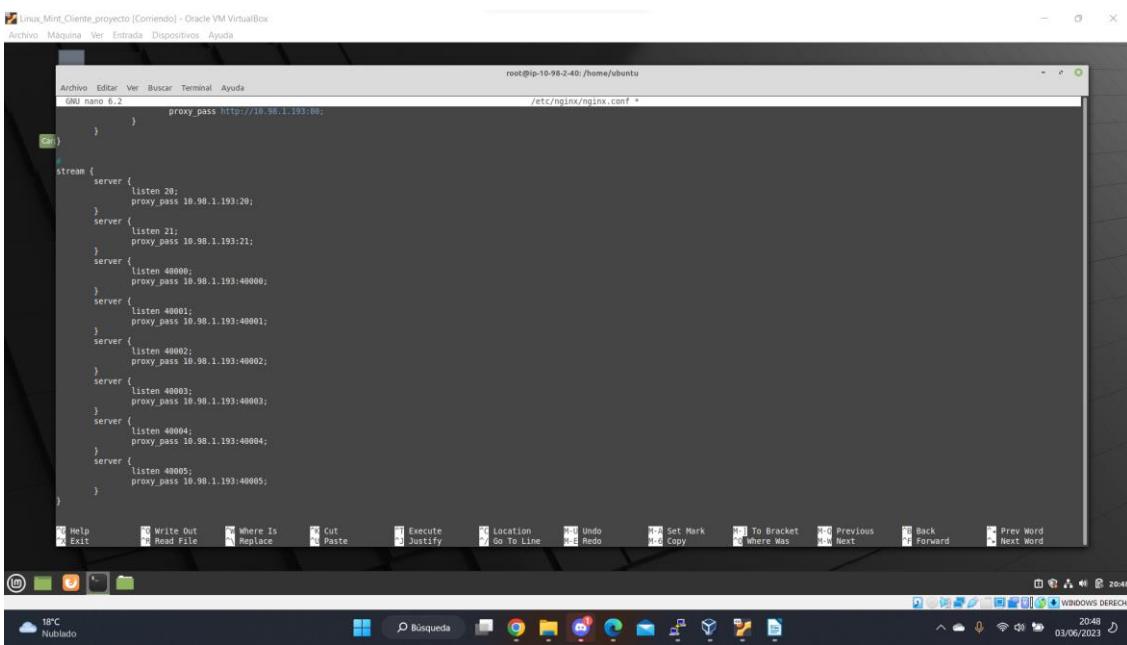
## Dockerización de un servidor completo en Cloud.

Ya podamos pasar a la construcción de la imagen.



Por último, la etiquetamos y la subimos a Docker Hub como hicimos en el manual anterior.

Ahora, nos conectamos al bastión y editamos el fichero de configuración de Nginx para añadir las reglas de reenvío al proxy inverso.



Se debe recordar reiniciar el servicio nginx para aplicar las nuevas reglas.

Por último, configuraremos los grupos de seguridad de las instancias, añadiendo en las reglas de entrada los puertos FTP con nuestra IP pública como Origen. Empezamos por el bastión.

| ID de la regla del grupo de seguridad | Tipo              | Información | Protocolo | Intervalo de puertos | Origen    | Información    | Descripción: opcional       |                           |
|---------------------------------------|-------------------|-------------|-----------|----------------------|-----------|----------------|-----------------------------|---------------------------|
| sgr-0887aab60be15c594                 | SSH               |             | TCP       | 22                   | Person... | 94.73.58.49/32 | Conexion_SSH                | <button>Eliminar</button> |
| sgr-00ae64c2446e9c741                 | HTTP              |             | TCP       | 80                   | Person... | 94.73.58.49/32 | HTTP_IP_publica_Gerardo     | <button>Eliminar</button> |
| sgr-047d7ff3f3934fefa5a               | HTTPS             |             | TCP       | 443                  | Person... | 94.73.58.49/32 | HTTPS_IP_publica_Gerardo    | <button>Eliminar</button> |
| -                                     | TCP personalizado |             | TCP       | 20 - 21              | Person... | 94.73.58.49/32 | FTP_IP_publica_Gerardo      | <button>Eliminar</button> |
| -                                     | TCP personalizado |             | TCP       | 40000 - 40           | Person... | 94.73.58.49/32 | FTP_PASV_IP_publica_Gerardo | <button>Eliminar</button> |

Y después el grupo del servidor. En origen indicamos la IP interna del bastión. En ambos grupos las reglas de salida se mantienen igual.

| ID de la regla del grupo de seguridad | Tipo              | Información | Protocolo | Intervalo de puertos | Origen    | Información   | Descripción: opcional |                           |
|---------------------------------------|-------------------|-------------|-----------|----------------------|-----------|---------------|-----------------------|---------------------------|
| sgr-0924d2c74a4012890                 | HTTPS             |             | TCP       | 443                  | Person... | 10.98.2.40/32 | HTTPS_bastion         | <button>Eliminar</button> |
| sgr-01b7dc59e7c22f5e4                 | HTTP              |             | TCP       | 80                   | Person... | 10.98.2.40/32 | HTTP_bastion          | <button>Eliminar</button> |
| sgr-03fc2a16e672c416d                 | SSH               |             | TCP       | 22                   | Person... | 10.98.2.40/32 | SSH_bastion           | <button>Eliminar</button> |
| -                                     | TCP personalizado |             | TCP       | 20 - 21              | Person... | 10.98.2.40/32 | FTP_bastion           | <button>Eliminar</button> |
| -                                     | TCP personalizado |             | TCP       | 40000 - 40           | Person... | 10.98.2.40/32 | FTP_PASV_bastion      | <button>Eliminar</button> |

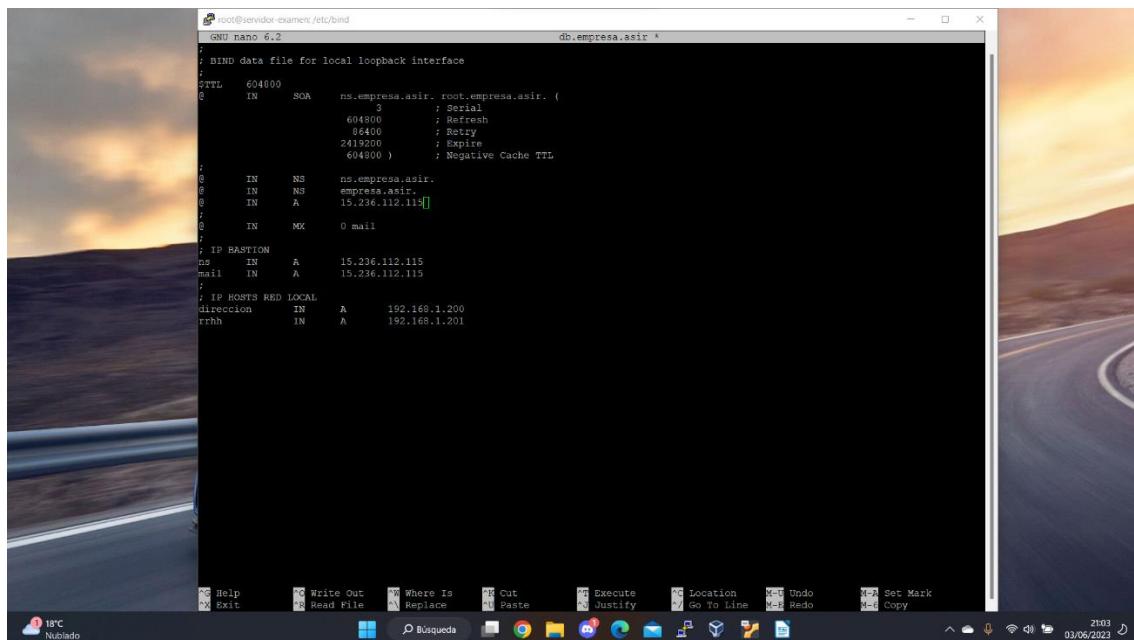
Queda así todo lo relacionado con este servicio listo para la prueba final en el servidor. Pulsa [aquí](#) para volver arriba.

## 10.10 Anexo 10: Manual DNS y correo electrónico.

Como en el anterior manual, empezamos por instalar los servicios en local para hacernos con los ficheros de configuración. Los paquetes que instalaremos serán: bind9, para el DNS, y postfix, dovecot-core, dovecot-imapd y dovecot-pop3d, para el correo electrónico.

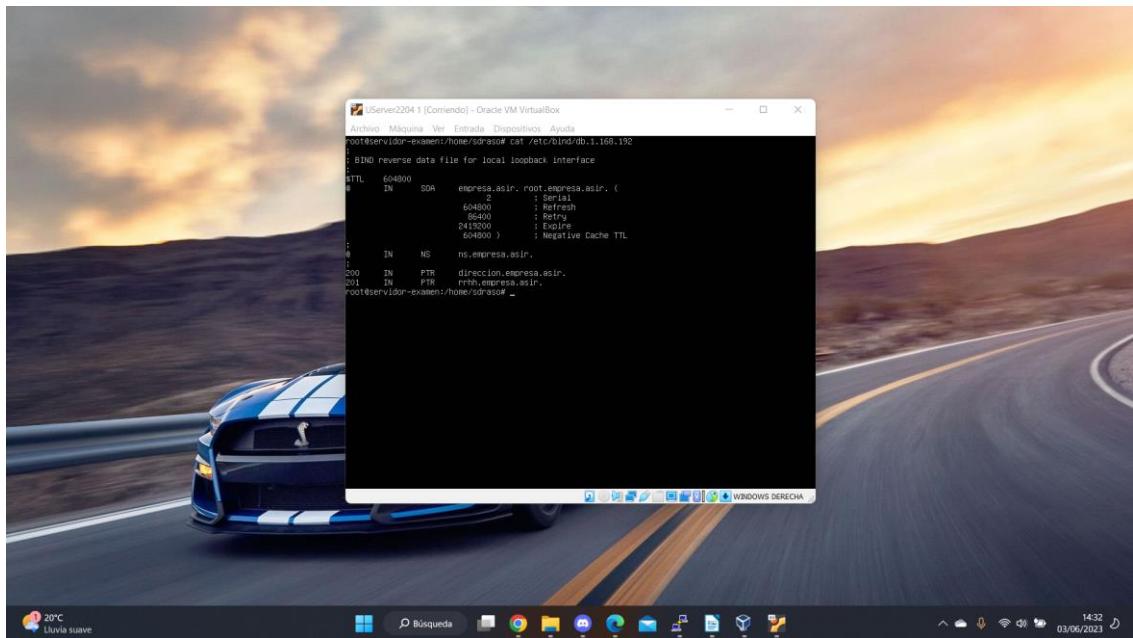
Empecemos por la configuración de los ficheros DNS. Crearemos una zona directa y tres indirectas. Las indirectas harán referencia a la red interna donde se sitúa el servidor, a la red local de la teórica empresa y a una creada para resolver la IP pública del bastión. También se editarán los ficheros named.conf.local y named.conf.options.

Empecemos por el fichero de zona directa (db.empresa.asir).

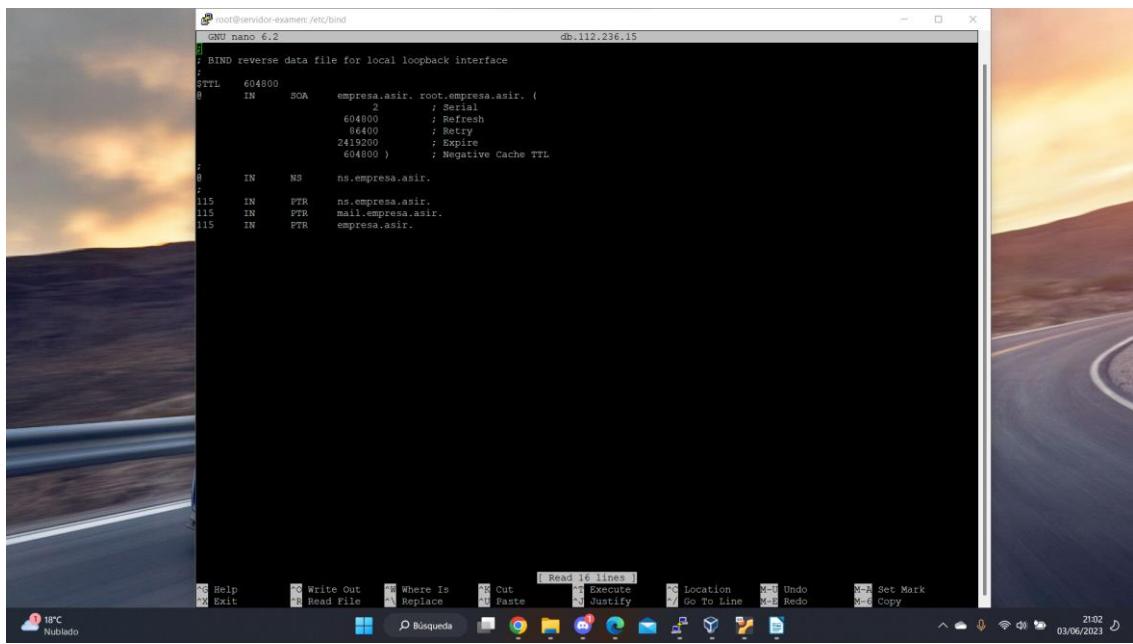


```
root@servidor-examen: /etc/bind
GNU nano 6.2                               db.empresa.asir *
; BIND data file for local loopback interface
;
;TTL    604800
$TTL   604800
@      IN    SOA    ns.empresa.asir. root.empresa.asir. (
                      3           ; Serial
                      604800     ; Refresh
                      86400      ; Retry
                     2419200   ; Expire
                     604800 )   ; Negative Cache TTL
;
@      IN    NS     ns.empresa.asir.
@      IN    NS     empresa.asir.
@      IN    A      15.236.112.115
@      IN    MX    0 mail
;
; IP BASTION
ns      IN    A      15.236.112.115
mail    IN    A      15.236.112.115
;
; IP HOSTS RED LOCAL
direccion IN    A      192.168.1.200
rrhh    IN    A      192.168.1.201
```

Sigamos con las zonas inversas. Primero el de la red local de la empresa (db.1.168.192):

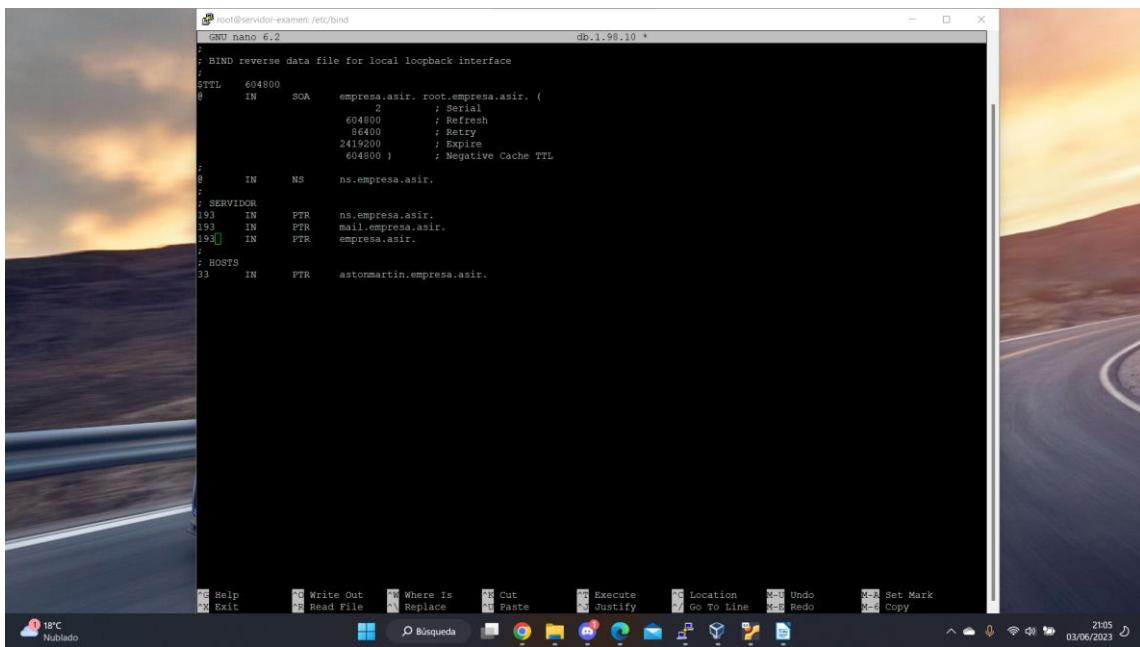


A continuación, la zona inversa creada para resolver la IP externa del bastión (db.112.236.15).



## Dockerización de un servidor completo en Cloud.

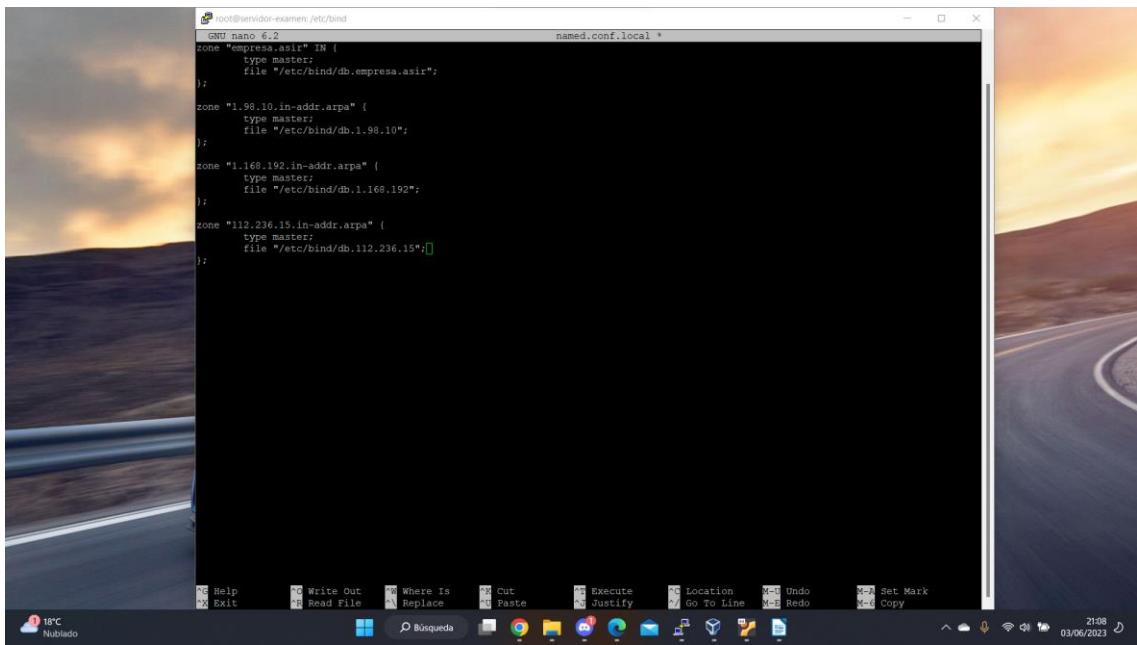
Terminamos con los ficheros de zona con la zona inversa de la subred privada en la que se ubica el servidor:



```
root@servidor-examen: /etc/bind
GNU nano 6.2                               db.1.98.10 *
;
; BIND reverse data file for local loopback interface
;
;TTL    604800
$TTL 604800
@      IN  SOA   empresa.asir. root.empresa.asir. (
                      2           ; Serial
                     604800      ; Refresh
                      86400       ; Retry
                     2419200     ; Expire
                     604800 )    ; Negative Cache TTL
;
;       IN  NS   ns.empresa.asir.
;
; SERVIDOR
193  IN  PTR   ns.empresa.asir.
193  IN  PTR   mail.empresa.asir.
193  IN  PTR   empresa.asir.
;
; HOSTS
39   IN  PTR   astonmartin.empresa.asir.

GNU nano 6.2                               db.1.98.10 *
Help Exit  Write Out Read File Where Is Replace Cut Execute Justify Location Go To Line Undo Redo Set Mark Copy
18°C Nublado  Búsqueda  21:05 03/06/2023
```

Editamos ahora named.conf.local para añadir las zonas:



```
root@servidor-examen: /etc/bind
GNU nano 6.2                               named.conf.local *
zone "empresa.asir" IN {
    type master;
    file "/etc/bind/db.empresa.asir";
};

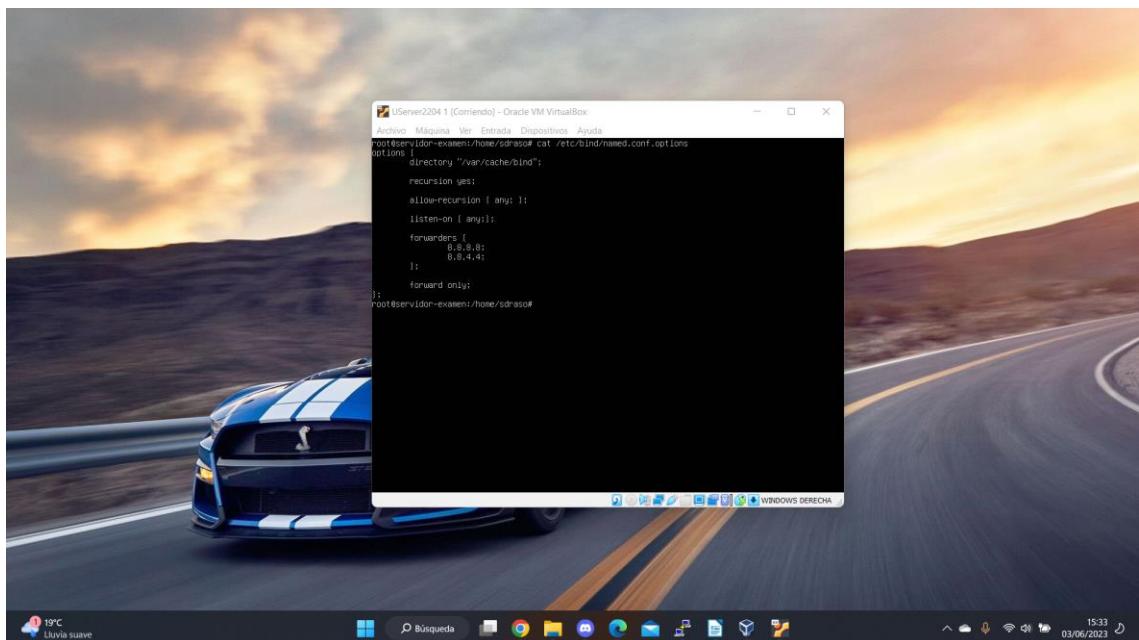
zone "1.98.10.in-addr.arpa" {
    type master;
    file "/etc/bind/db.1.98.10";
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.1.168.192";
};

zone "112.236.15.in-addr.arpa" {
    type master;
    file "/etc/bind/db.112.236.15";}
;

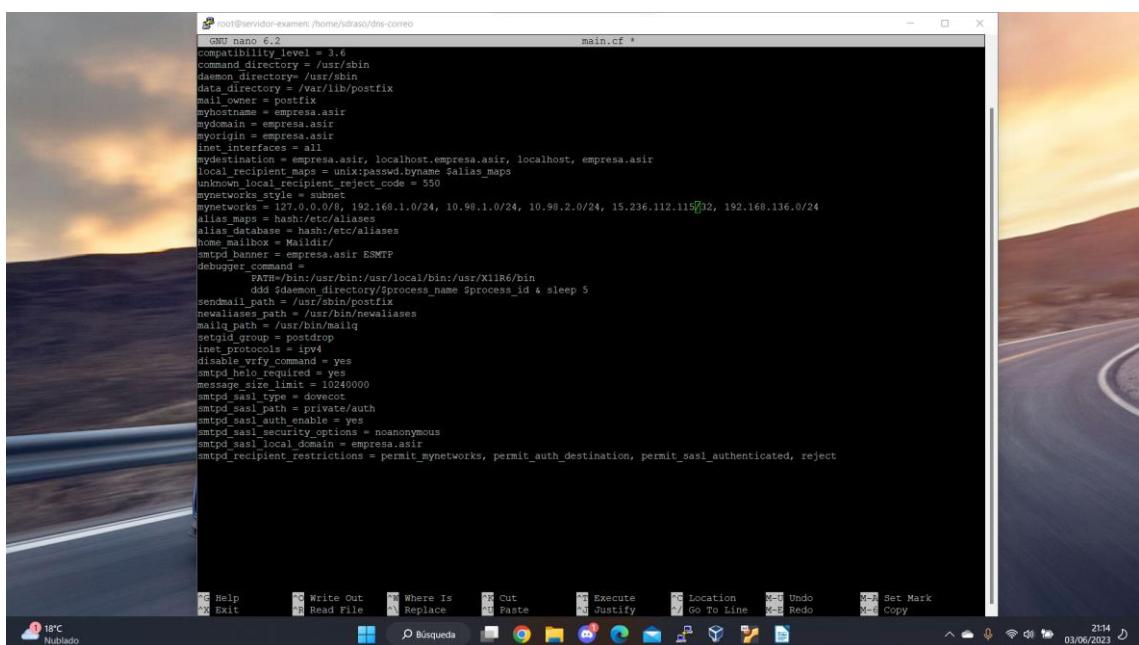
GNU nano 6.2                               named.conf.local *
Help Exit  Write Out Read File Where Is Replace Cut Execute Justify Location Go To Line Undo Redo Set Mark Copy
18°C Nublado  Búsqueda  21:08 03/06/2023
```

Y por último, colocamos la configuración elegida para el servidor DNS en named.conf.options. Se configura el servidor como reenviador para poder resolver tanto nombres internos de la propia empresa como externos.



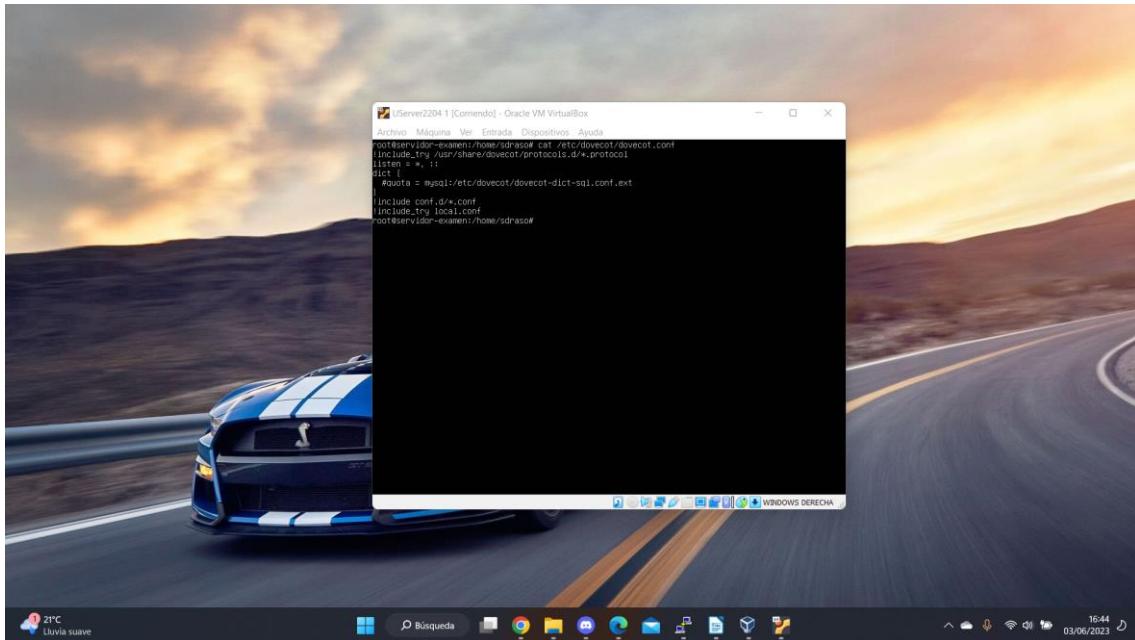
Tras comprobar la sintaxis y chequear que esta configuración permite que los servicios se ejecuten, pasamos a los ficheros de correo electrónico.

Comenzamos por el main.cf de la configuración de postfix, en el que añadimos nuestro dominio y las redes que manejamos.



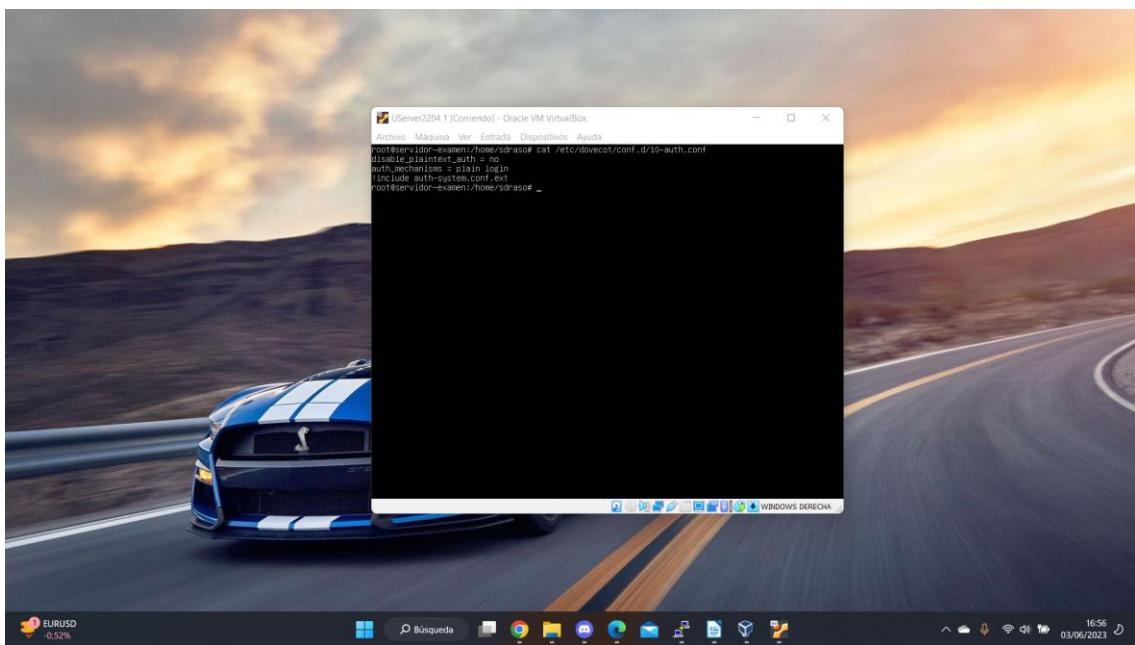
## Dockerización de un servidor completo en Cloud.

Pasamos a los ficheros de configuración de dovecot. Editamos dovecot.conf.

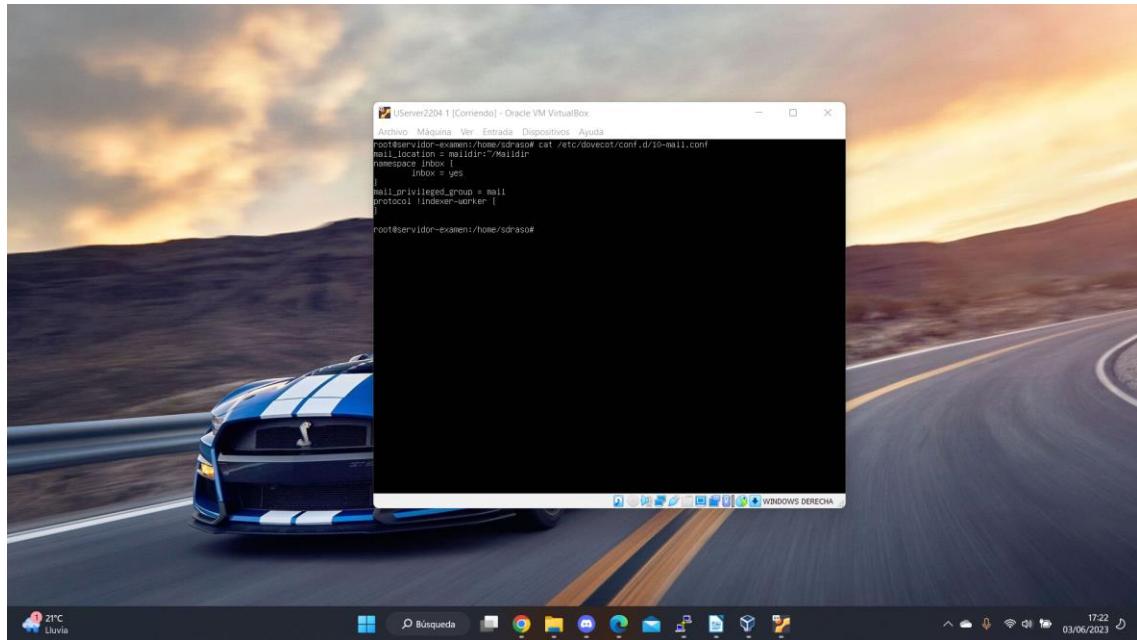


Seguimos con '10-auth.conf', '10-mail.conf' y '10-master.conf', situados en el directorio conf.d.

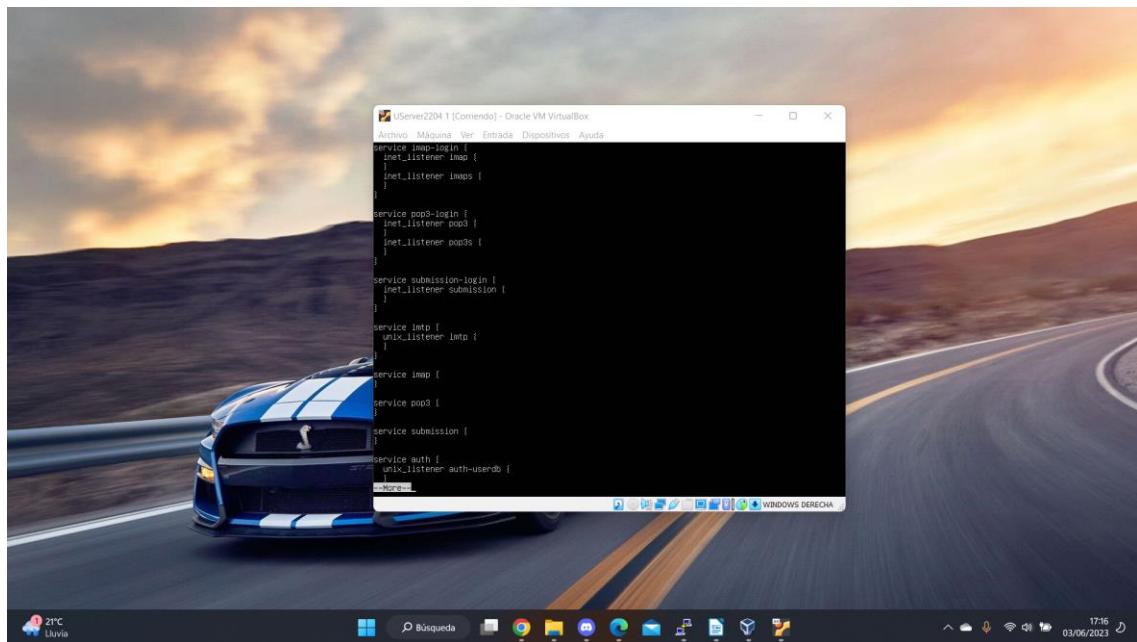
### 10-auth.conf

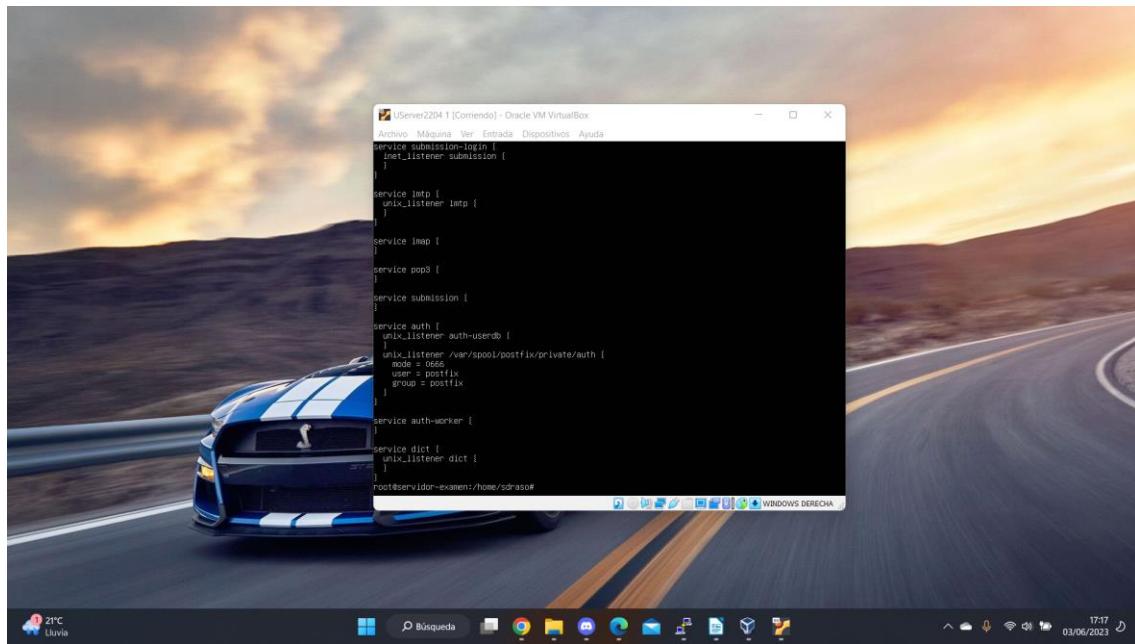
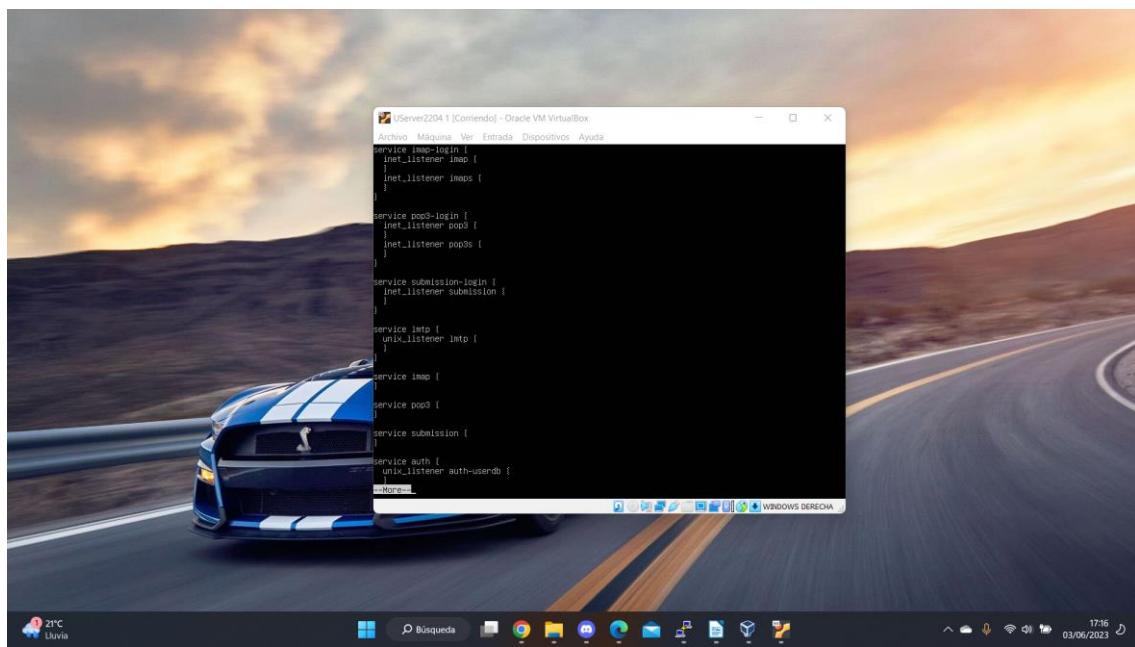


### 10-mail.conf

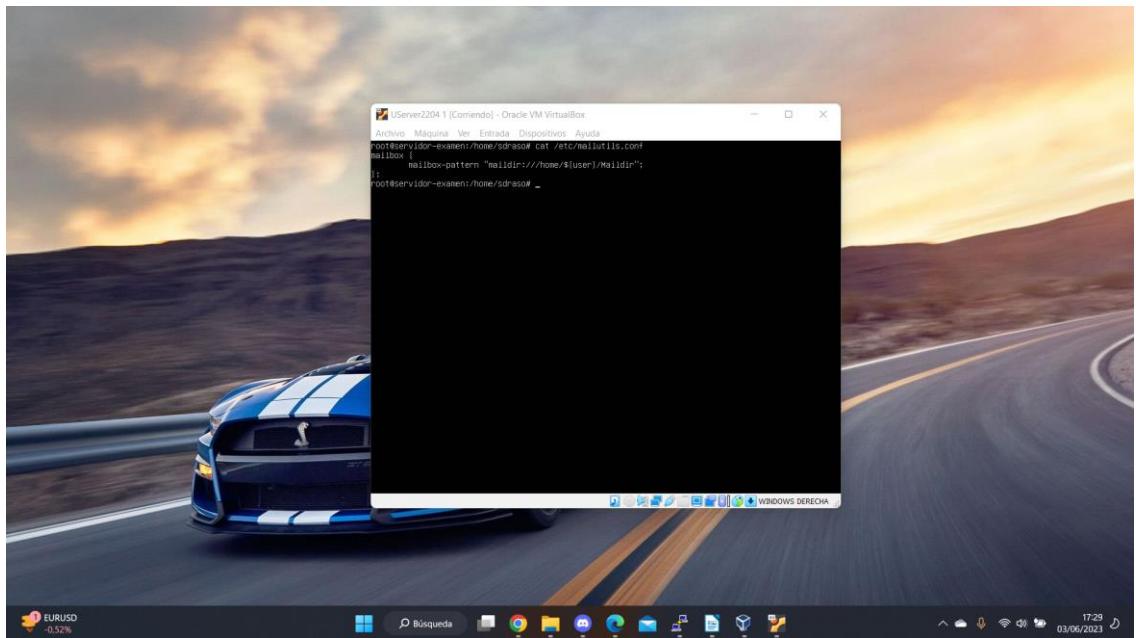


### 10-master.conf



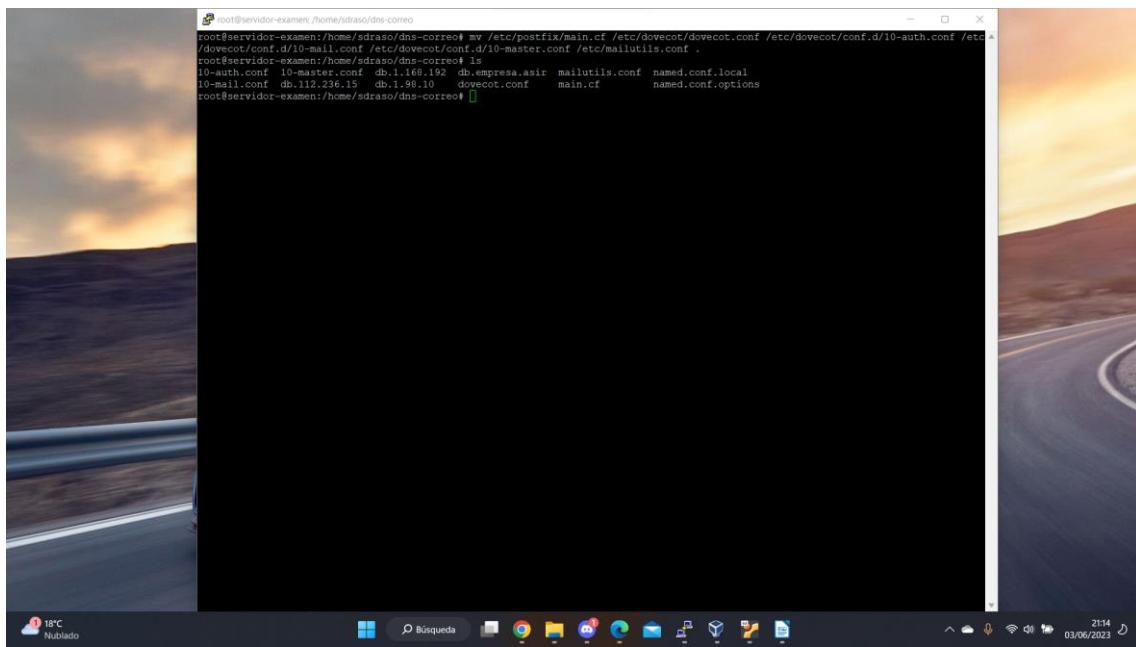


Finalmente, creamos el fichero de configuración de mailutils (mailutils.conf).



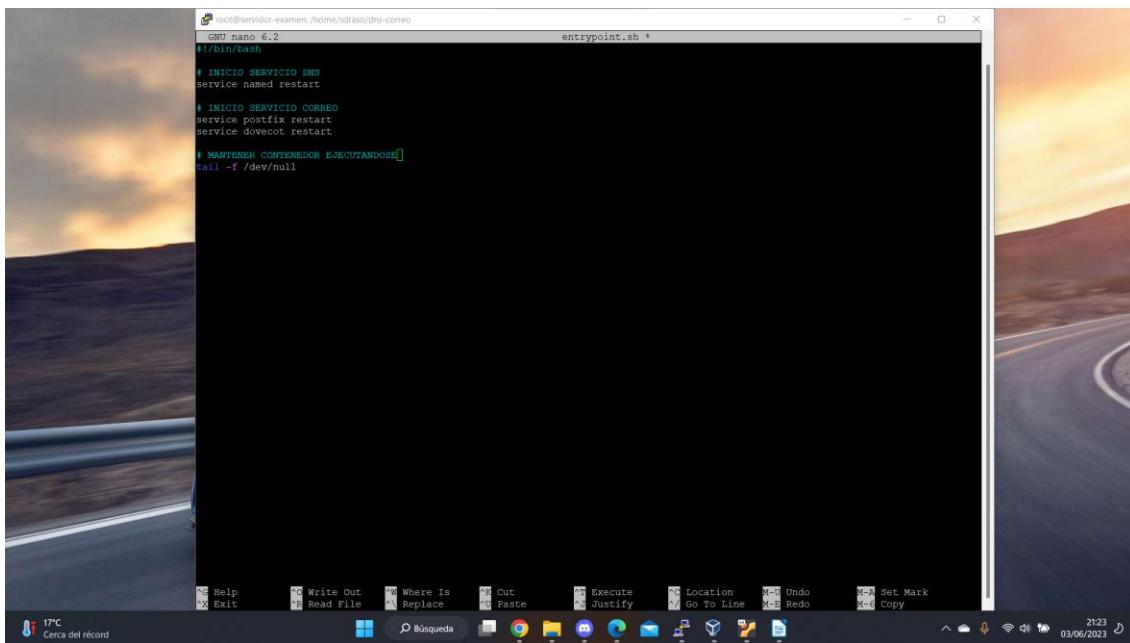
Comprobamos su sintaxis y que los servicios aceptan la configuración. Si es así, podemos pasar a la imagen.

Ahora que tenemos todo lo necesario, creamos la carpeta dns-correo para trabajar en ella. Movemos a ella todos los archivos de configuración mencionados en el manual.



## Dockerización de un servidor completo en Cloud.

Antes de crear el Dockerfile, vamos a seguir el mismo método que en el anterior manual para reiniciar los servicios. Creamos un script llamado 'entrypoint.sh' con las instrucciones pertinentes:



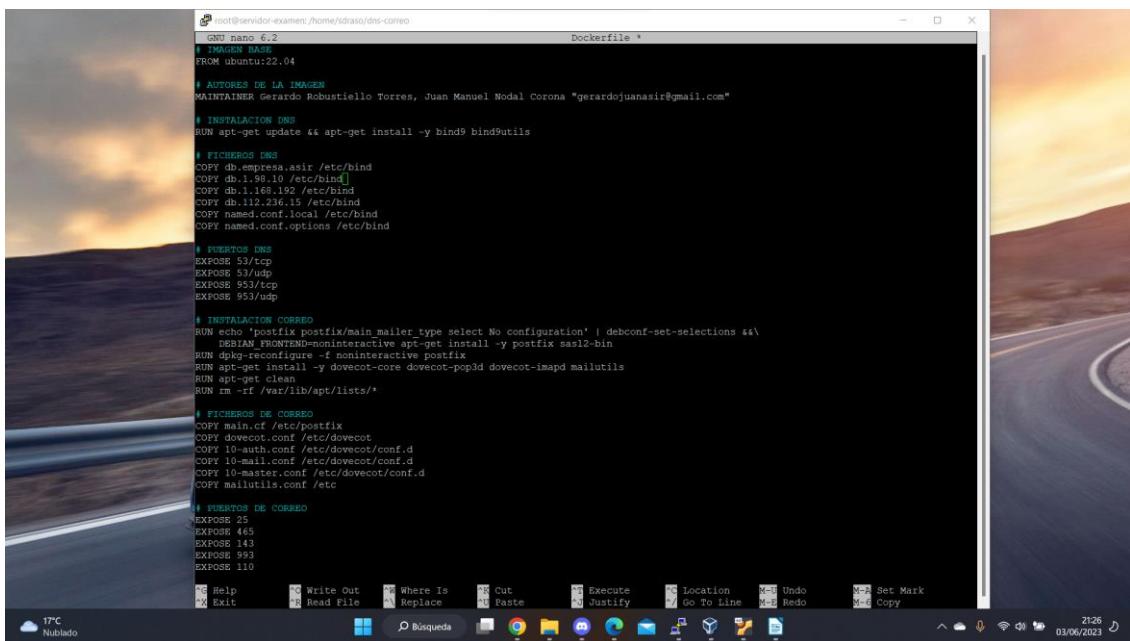
```
root@servidor-examen: /home/sdraso/dns-correo
GNU nano 6.2                               entrypoint.sh
#!/bin/bash

# INICIO SERVICIO DNS
service named restart

# INICIO SERVICIO CORREO
service postfix restart
service dovecot restart

# MANTENER CONTENEDOR EJECUTANDOSE
tail -f /dev/null
```

Creamos ya el Dockefile de la imagen. Las instrucciones serán las siguientes: Ubuntu 22.04 como imagen base, instalación de paquetes (bind9, bind9utils, postfix, sasl2-bin, dovecot-core, dovecot-imapd, dovecot-pop3d y mailutils). Como la instalación de dovecot requiere de la intervención del usuario para llevarla a cabo, se han añadido instrucciones para efectuar una instalación desatendida. También se han copiado los archivos, se han abierto los puertos y se han creado dos usuarios para probar el servicio de correo. Se incluye la ejecución del script de inicio de servicios.



```
root@servidor-examen: /home/sdraso/dns-correo
GNU nano 6.2                               Dockerfile
# IMAGEN BASE
FROM ubuntu:22.04

# AUTORES DE LA IMAGEN
MAINTAINER Gerardo Robustielo Torres, Juan Manuel Nodal Corona "gerardojuanasir@gmail.com"

# INSTALACION DNS
RUN apt-get update && apt-get install -y bind9 bind9utils

# FICHEROS DNS
COPY db.empresa.asn /etc/bind
COPY db.1.1.1.1.169.192 /etc/bind
COPY db.112.236.15 /etc/bind
COPY named.conf.local /etc/bind
COPY named.conf.options /etc/bind

# PUERTOS DNS
EXPOSE 53/tcp
EXPOSE 53/udp
EXPOSE 953/tcp
EXPOSE 953/udp

# INSTALACION CORREO
RUN echo 'postfix postfix/main_mailer_type select No configuration' | debconf-set-selections &&
DEBIAN_FRONTEND=noninteractive apt-get install -y postfix sasl2-bin
RUN dpkg-reconfigure -f noninteractive postfix
RUN apt-get install -y dovecot-core dovecot-pop3d dovecot-imapd mailutils
RUN apt-get clean
RUN rm -rf /var/lib/apt/lists/*

# FICHEROS DE CORREO
COPY main.cf /etc/postfix
COPY dovecot.conf /etc/dovecot
COPY 10-auth.conf /etc/dovecot/conf.d
COPY 10-mail.conf /etc/dovecot/conf.d
COPY 10-master.conf /etc/dovecot/conf.d
COPY mailutils.conf /etc

# PUERTOS DE CORREO
EXPOSE 25
EXPOSE 465
EXPOSE 113
EXPOSE 993
EXPOSE 110
```

```
#!/bin/bash
# Dockerfile for a Postfix and Dovecot container

# Base image
FROM alpine:latest

# Environment variables
ENV TZ=UTC
ENV POSTFIX_RELAYHOST="127.0.0.1"
ENV DOVECOT_RELAYHOST="127.0.0.1"
ENV DOVECOT_USER="dovecot"
ENV DOVECOT_PASSWORD="dovecot"

# Install dependencies
RUN apk add --no-cache curl ca-certificates
RUN curl -s https://dl.bintray.com/dovecot/dovecot-stable/alpine/2.9.15/dovecot-2.9.15.tar.gz | tar xz
RUN cd dovecot-2.9.15
RUN ./configure --prefix=/usr/local/dovecot
RUN make
RUN make install
RUN rm -rf /var/lib/apt/lists/*
RUN rm -rf /var/lib/dovecot

# Copy configuration files
COPY dovecot.conf /etc/dovecot
COPY 10-auth.conf /etc/dovecot/conf.d
COPY 10-mail.conf /etc/dovecot/conf.d
COPY 10-master.conf /etc/dovecot/conf.d
COPY mailutils.conf /etc

# Set environment variables
ENV DOVECOT_RELAYHOST="127.0.0.1"
ENV DOVECOT_USER="dovecot"
ENV DOVECOT_PASSWORD="dovecot"

# Expose ports
EXPOSE 25
EXPOSE 465
EXPOSE 143
EXPOSE 587
EXPOSE 110
EXPOSE 995

# Configuration extra
RUN newaliases
ENV DOVECOT_MAILDIR=~/mail
ENV DOVECOT_PASSWORD=dovecot
ENV DOVECOT_USER=dovecot
ENV DOVECOT_RELAYHOST=127.0.0.1
ENV DOVECOT_RELAYPORT=25
ENV DOVECOT_IMAPD_PORT=143
ENV DOVECOT_IMAPD_SSL_PORT=993
ENV DOVECOT_POP3D_PORT=110
ENV DOVECOT_POP3D_SSL_PORT=587

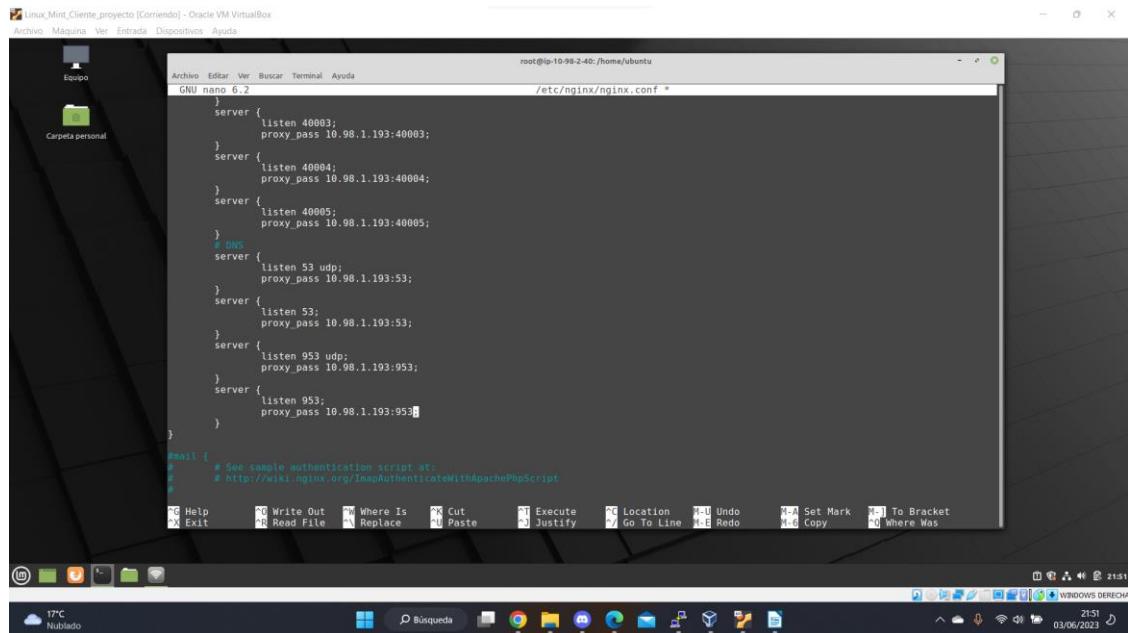
# Start service
COPY entrypoint.sh /entrypoint.sh
RUN chmod a+x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

Procedemos a construir la imagen.

Posteriormente, como en anteriores manuales, la etiquetamos y la subimos a Docker Hub.

## Dockerización de un servidor completo en Cloud.

El siguiente paso será la configuración del proxy inverso del bastión para permitir el reenvío de las peticiones de los clientes. Empezamos por las reglas para el DNS.



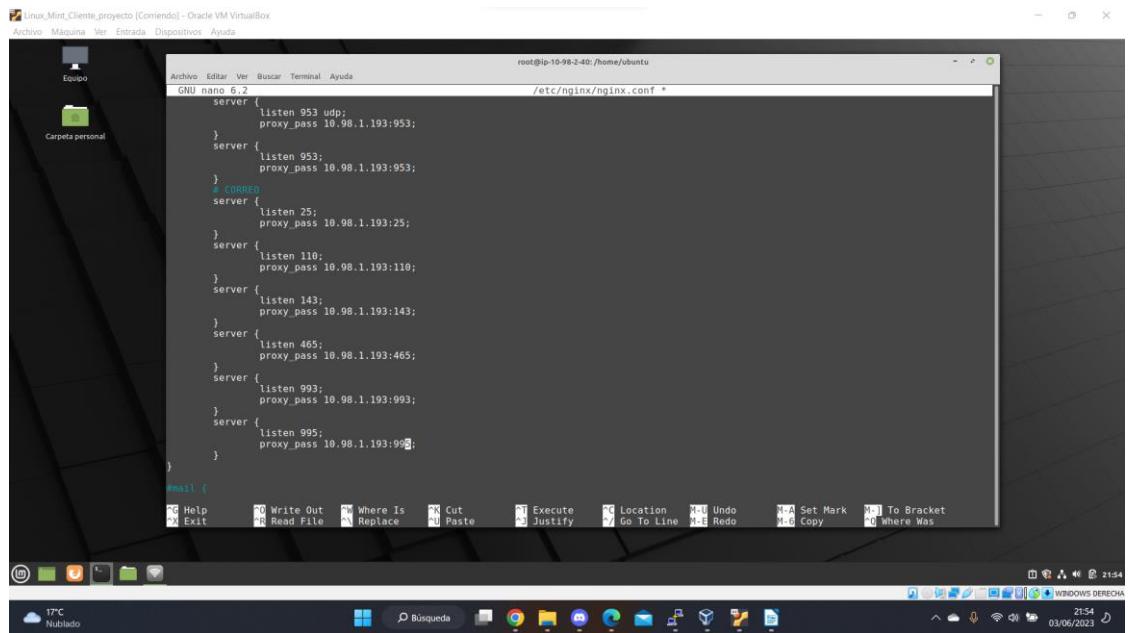
```
root@ip-10-98-2-40:/home/ubuntu
GNU nano 6.2
server {
    listen 40003;
    proxy_pass 10.98.1.193:40003;
}
server {
    listen 40004;
    proxy_pass 10.98.1.193:40004;
}
server {
    listen 40005;
    proxy_pass 10.98.1.193:40005;
}
# DNS
server {
    listen 53 udp;
    proxy_pass 10.98.1.193:53;
}
server {
    listen 53;
    proxy_pass 10.98.1.193:53;
}
server {
    listen 953 udp;
    proxy_pass 10.98.1.193:953;
}
server {
    listen 953;
    proxy_pass 10.98.1.193:953;
}

mail {
    # See sample authentication script at:
    # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
}

# Help   W Write Out  W Where Is  C Cut  E Execute  F Location  M-U Undo  M-A Set Mark  M-T To Bracket
# Exit  R Read File  R Replace  P Paste  J Justify  G Go To Line  M-R Redo  M-C Copy  M-W Where Was

```

Y a continuación ponemos las de correo.



```
root@ip-10-98-2-40:/home/ubuntu
GNU nano 6.2
server {
    listen 953 udp;
    proxy_pass 10.98.1.193:953;
}
server {
    listen 953;
    proxy_pass 10.98.1.193:953;
}
# CORREO
server {
    listen 25;
    proxy_pass 10.98.1.193:25;
}
server {
    listen 110;
    proxy_pass 10.98.1.193:110;
}
server {
    listen 143;
    proxy_pass 10.98.1.193:143;
}
server {
    listen 465;
    proxy_pass 10.98.1.193:465;
}
server {
    listen 993;
    proxy_pass 10.98.1.193:993;
}
server {
    listen 995;
    proxy_pass 10.98.1.193:995;
}

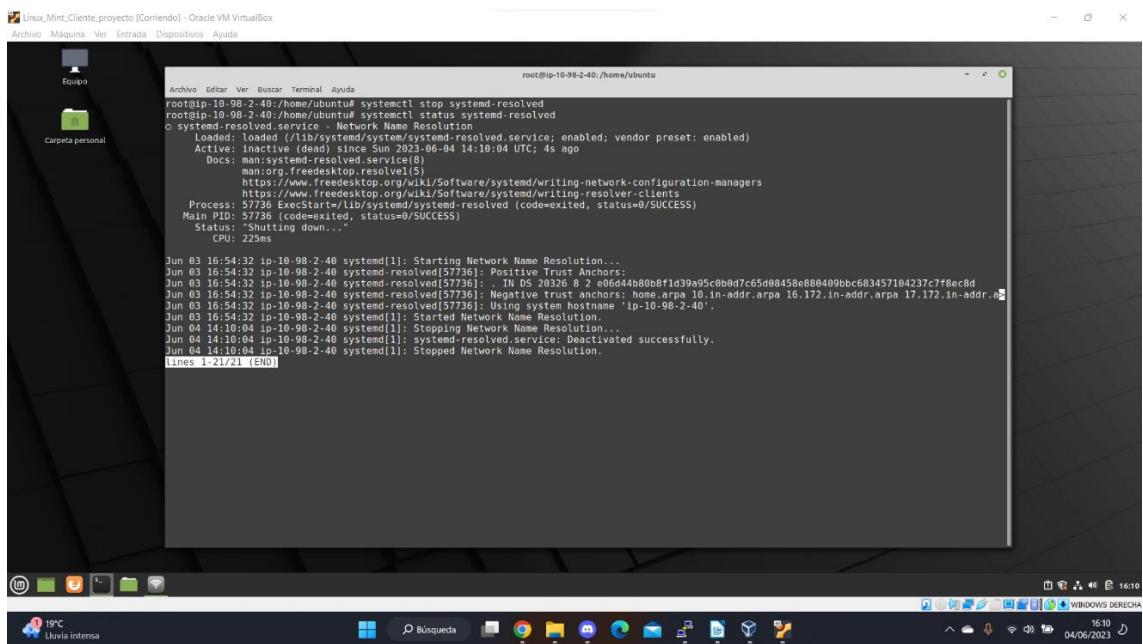
mail {
    # See sample authentication script at:
    # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
}

# Help   W Write Out  W Where Is  C Cut  E Execute  F Location  M-U Undo  M-A Set Mark  M-T To Bracket
# Exit  R Read File  R Replace  P Paste  J Justify  G Go To Line  M-R Redo  M-C Copy  M-W Where Was

```

Se debe recordar reiniciar el servicio nginx para aplicar las nuevas reglas.

Aprovechando que estamos en el bastión, vamos a desactivar el servicio ‘systemd-resolved’, que puede interferir con el servicio DNS que dará el servidor.



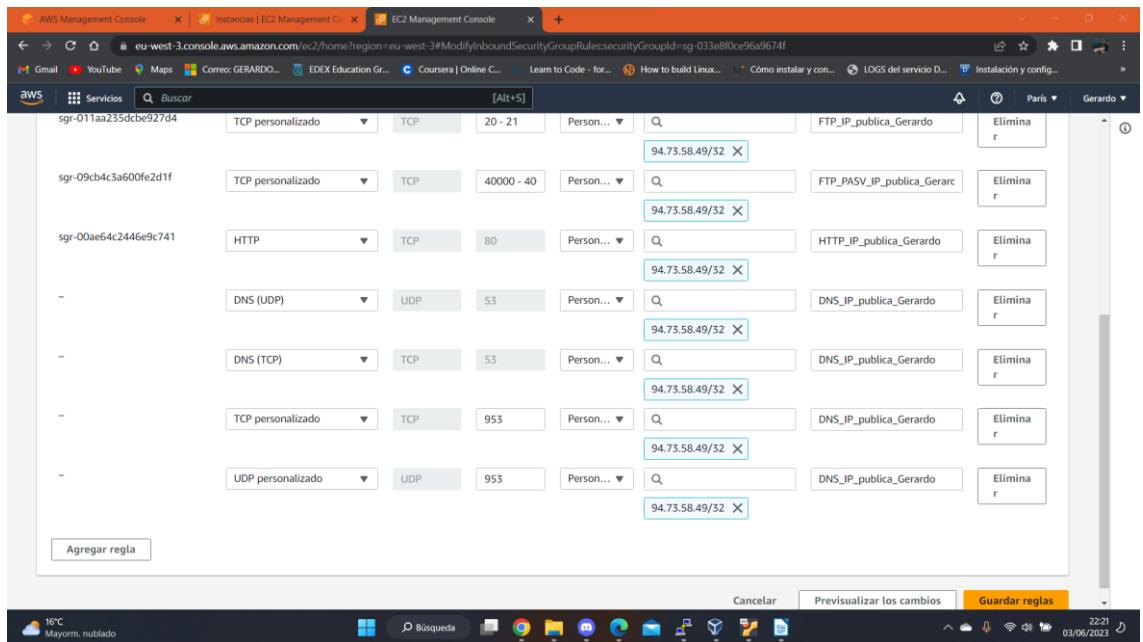
```
root@ip-10-98-2-40:/home/ubuntu
root@ip-10-98-2-40:/home/ubuntu# systemctl stop systemd-resolved
root@ip-10-98-2-40:/home/ubuntu# systemctl status systemd-resolved
● systemd-resolved.service - Network Name Resolution
   Loaded: loaded (/lib/systemd/system/systemd-resolved.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Sun 2023-06-04 14:10:04 UTC; 4s ago
     Docs: man(5 systemd-resolved)
           https://www.freedesktop.org/wiki/Software/systemd/writing-network-configuration-managers
           https://www.freedesktop.org/wiki/Software/systemd/writing-resolver-clients
   Process: 57738 ExecStart=/lib/systemd/systemd-resolved (code=exited, status=0/SUCCESS)
   Main PID: 57738 (code=exited, status=0/SUCCESS)
      Status: Shutting down...
        CPU: 22ms

Jun 03 16:54:32 ip-10-98-2-40 systemd[1]: Starting Network Name Resolution...
Jun 03 16:54:32 ip-10-98-2-40 systemd-resolved[57736]: Positive Trust Anchors:
Jun 03 16:54:32 ip-10-98-2-40 systemd-resolved[57736]:   TN DS 20326 8 2 006d4ab80b8f1d39a95c0b0d7c65d08458e880409bbc683457104237c7+8e8d
Jun 03 16:54:32 ip-10-98-2-40 systemd-resolved[57736]: Negative trust anchors: home.arp 10.in-addr.arp 16.172.in-addr.arp 17.172.in-addr.arp
Jun 03 16:54:32 ip-10-98-2-40 systemd-resolved[57736]: Using system hostname 'ip-10-98-2-40'.
Jun 03 16:54:32 ip-10-98-2-40 systemd[1]: Started Network Name Resolution.
Jun 04 14:10:04 ip-10-98-2-40 systemd[1]: Stopping Network Name Resolution...
Jun 04 14:10:04 ip-10-98-2-40 systemd[1]: systemd-resolved.service: Deactivated successfully.
Jun 04 14:10:04 ip-10-98-2-40 systemd[1]: Stopped Network Name Resolution.

[lines 1-21/21 (END)]
```

Por último, crearemos las reglas de los grupos de seguridad que permitirán el tráfico por los puertos adecuados. Empezamos por el grupo de seguridad del bastión. Recordemos que en ‘Origen’ tenemos la IP externa de nuestro router.

Primero las reglas de entrada para DNS.



| Regla                  | Tipo              | Puerto     | Protocolo | Origen                     | Destino                       |
|------------------------|-------------------|------------|-----------|----------------------------|-------------------------------|
| sgr-011aa235dcbe5927d4 | TCP personalizado | 20 - 21    | TCP       | Person... (94.73.58.49/32) | FTP_IP_publica_Gerardo        |
| sgr-09cb4c3a600fe2d1f  | TCP personalizado | 40000 - 40 | TCP       | Person... (94.73.58.49/32) | FTP_PASV_IP_publica_Gerard... |
| sgr-00ae64c2446e9c741  | HTTP              | 80         | TCP       | Person... (94.73.58.49/32) | HTTP_IP_publica_Gerardo       |
| -                      | DNS (UDP)         | 53         | UDP       | Person... (94.73.58.49/32) | DNS_IP_publica_Gerardo        |
| -                      | DNS (TCP)         | 53         | TCP       | Person... (94.73.58.49/32) | DNS_IP_publica_Gerardo        |
| -                      | TCP personalizado | 953        | TCP       | Person... (94.73.58.49/32) | DNS_IP_publica_Gerardo        |
| -                      | UDP personalizado | 953        | UDP       | Person... (94.73.58.49/32) | DNS_IP_publica_Gerardo        |

## Dockerización de un servidor completo en Cloud.

Y ahora las de correo electrónico:

The screenshot shows the AWS Management Console with the EC2 Management Console tab selected. A search bar at the top has "[Alt+S]" entered. Below it is a table of security group rules:

| Regla                 | Tipo              | Protocolo | Puerto | Origen      | Destino        | Opciones |
|-----------------------|-------------------|-----------|--------|-------------|----------------|----------|
| sgr-0e57daecce0fbcc6b | UDP personalizado | UDP       | 953    | Person... ▾ | 94.73.58.49/32 | Eliminar |
| sgr-02fc64d0ae37d1d0f | DNS (UDP)         | UDP       | 53     | Person... ▾ | 94.73.58.49/32 | Eliminar |
| -                     | SMTP              | TCP       | 25     | Person... ▾ | 94.73.58.49/32 | Eliminar |
| -                     | IMAP              | TCP       | 143    | Person... ▾ | 94.73.58.49/32 | Eliminar |
| -                     | POP3              | TCP       | 110    | Person... ▾ | 94.73.58.49/32 | Eliminar |
| -                     | IMAPS             | TCP       | 993    | Person... ▾ | 94.73.58.49/32 | Eliminar |
| -                     | SMTPS             | TCP       | 465    | Person... ▾ | 94.73.58.49/32 | Eliminar |
| -                     | POP3S             | TCP       | 995    | Person... ▾ | 94.73.58.49/32 | Eliminar |

Por último, configuramos las reglas de entrada del grupo del servidor. En 'Origen' colocamos la IP interna del bastión. Primero las reglas DNS:

The screenshot shows the AWS Management Console with the EC2 Management Console tab selected. A search bar at the top has "[Alt+S]" entered. Below it is a table of security group rules:

| Regla                  | Tipo              | Protocolo | Puerto     | Origen      | Destino       | Opciones |
|------------------------|-------------------|-----------|------------|-------------|---------------|----------|
| sgr-04010adebe550cd975 | TCP personalizado | TCP       | 20 - 21    | Person... ▾ | 10.98.2.40/32 | Eliminar |
| sgr-01b7dc59e7c22f5e4  | HTTP              | TCP       | 80         | Person... ▾ | 10.98.2.40/32 | Eliminar |
| sgr-07bfa8124a1d563ed  | TCP personalizado | TCP       | 40000 - 40 | Person... ▾ | 10.98.2.40/32 | Eliminar |
| sgr-03fc2a16e672c416d  | SSH               | TCP       | 22         | Person... ▾ | 10.98.2.40/32 | Eliminar |
| -                      | DNS (UDP)         | UDP       | 53         | Person... ▾ | 10.98.2.40/32 | Eliminar |
| -                      | DNS (TCP)         | TCP       | 53         | Person... ▾ | 10.98.2.40/32 | Eliminar |
| -                      | TCP personalizado | TCP       | 953        | Person... ▾ | 10.98.2.40/32 | Eliminar |
| -                      | UDP personalizado | UDP       | 953        | Person... ▾ | 10.98.2.40/32 | Eliminar |

Y ahora las de correo:

The screenshot shows the AWS Management Console with the EC2 Management Console tab selected. A security group named 'sgr-Uddcezb0rb54e4t25' is being configured. The inbound rules section lists the following entries:

| Protocolo         | Puerto | Origen      | Destino        | Opciones      |
|-------------------|--------|-------------|----------------|---------------|
| TCP personalizado | 953    | Personas... | DNS_bastion    | 10.98.2.40/32 |
| SSH               | 22     | Personas... | SSH_bastion    | 10.98.2.40/32 |
| SMTP              | 25     | Personas... | CORREO_bastion | 10.98.2.40/32 |
| SMTPS             | 465    | Personas... | CORREO_bastion | 10.98.2.40/32 |
| IMAP              | 143    | Personas... | CORREO_bastion | 10.98.2.40/32 |
| IMAPS             | 993    | Personas... | CORREO_bastion | 10.98.2.40/32 |
| POP3              | 110    | Personas... | CORREO_bastion | 10.98.2.40/32 |
| POP3S             | 995    | Personas... | CORREO_bastion | 10.98.2.40/32 |

En cuanto a las reglas de salida de ambos grupos, se mantienen igual. Pulsa [aquí](#) para volver arriba.

## 10.11 Anexo 11: Manual de despliegue de imágenes.

Vamos a crear los contenedores que ofrecerán los servicios prometidos por el servidor. Para ello, nos conectamos por SSH desde el bastión.

→ `ssh -i "clave_servidor.pem" ubuntu@10.98.1.193`

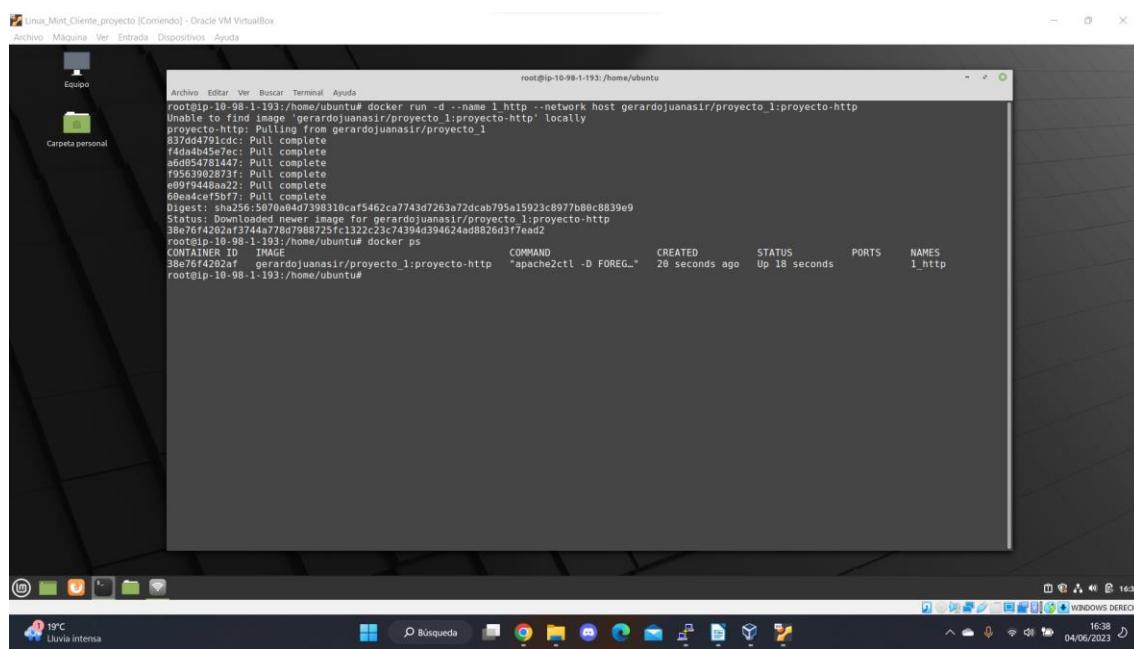
Siguiendo el [manual de instalación de Docker](#), lo instalamos en el servidor para poder desplegar las imágenes. Una vez lo tenemos listo, procedemos. Empezaremos con el contenedor del servicio HTTP. El comando de despliegue es el siguiente:

→ `docker run -d --name l_http --network host gerardojuanasir:proyecto_1:proyecto-http`

Donde ‘-d’ significa que el proceso se ejecuta en segundo plano, ‘--name’ indica el nombre del contenedor, ‘–network host’ indica que el contenedor utiliza la red de su host, es decir, el servidor y ‘gerardojuanasir:proyecto\_1:proyecto-http’ hace referencia al nombre de usuario en Docker Hub, el nombre del repositorio y la etiqueta que da nombre a la imagen.

Una vez desplegado, miramos los procesos docker para ver que el contenedor está funcionando:

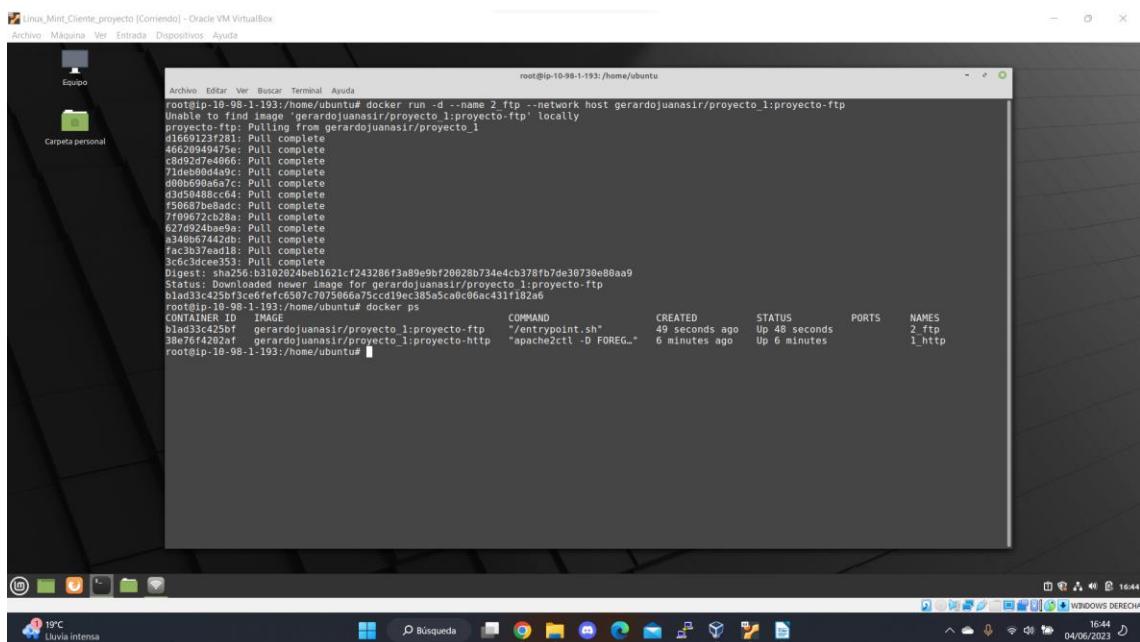
→ `docker ps`



The screenshot shows a terminal window titled 'root@ip-10-98-1-193:/home/ubuntu'. The window displays the following command and its output:

```
root@ip-10-98-1-193:/home/ubuntu# docker run -d --name l_http --network host gerardojuanasir:proyecto_1:proyecto-http
Unable to find image 'gerardojuanasir/proyecto_1:proyecto-http' locally
proyecto-http: Pulling from gerardojuanasir/proyecto_1
3370743f3377: Pulling fs layer
f4dab45e7ec: Pull complete
a6d8d4781447: Pull complete
f9563902873f: Pull complete
e9719448a229: Pull complete
60eaefc4f507: Pull complete
Digest: sha256:5070b04d7398310caf5462ca7743d7263a72dcab795a15923c8977b80r8839e9
Status: Downloaded newer image for gerardojuanasir:proyecto_1:proyecto-http
38e76f4202a73744a778d7988725fc132c23c74394d394624ad8826d3f7ead2
root@ip-10-98-1-193:/home/ubuntu# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
38e76f4202a7        gerardojuanasir/proyecto_1:proyecto-http   "apache2ctl -D FOREG..."   28 seconds ago    Up 18 seconds          l_http
root@ip-10-98-1-193:/home/ubuntu#
```

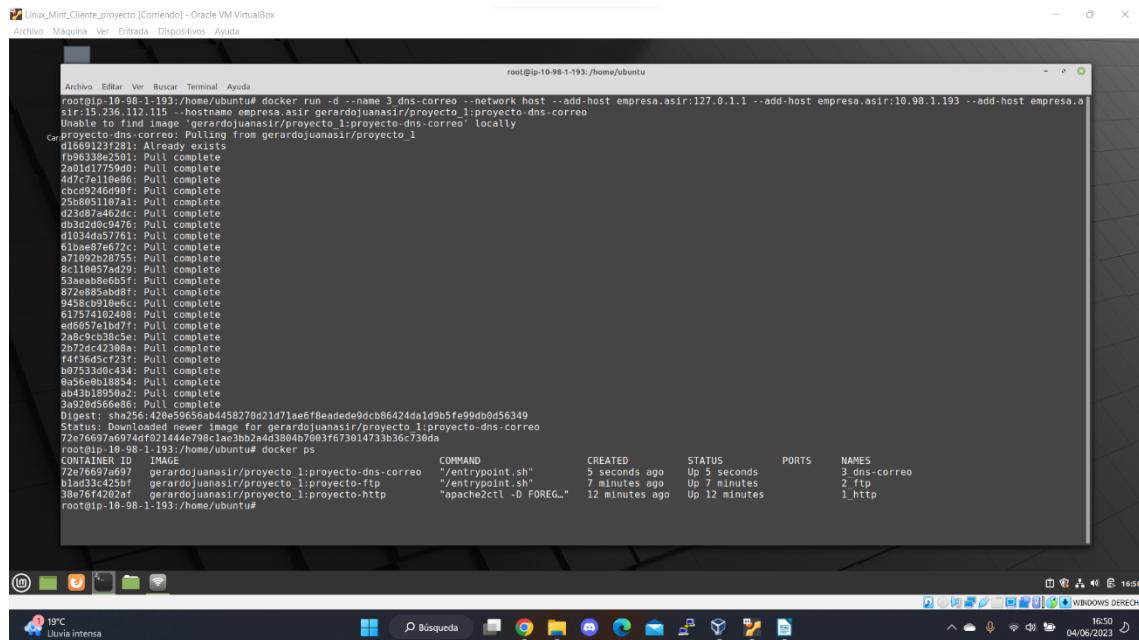
Seguimos la misma metodología para desplegar el contenedor FTP, en el que solo cambiamos el nombre del contenedor y la etiqueta de la imagen para usar la correcta. Volvemos a mirar los procesos Docker para comprobar que el despliegue ha sido satisfactorio.



Por último, vamos a desplegar el contenedor DNS y correo. Este requiere algo de configuración extra. En el comando se indica con '--hostname' el nombre de red que queremos que tenga el host (empresa.asir) y agregamos también tres entradas a su fichero '/etc/hosts' mediante la instrucción '--add-host'. Estas entradas hacen sirven para identificar al servidor con el nombre de dominio empresa.asir

## Dockerización de un servidor completo en Cloud.

→ `docker run -d --name 3_dns-correo --host network --hostname empresa.asir --add-host empresa.asir:127.0.1.1 --add-host empresa.asir:10.98.1.193 --add-host empresa.asir:15.236.112.115 gerardojuanasir/proyecto_1:projeto-dns-correo`



```
Linux_Mint_Cliente_proyecto [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
root@ip-10-98-1-193:/home/ubuntu
root@ip-10-98-1-193:/home/ubuntu# docker run -d --name 3_dns-correo --network host --hostname empresa.asir:127.0.1.1 --add-host empresa.asir:10.98.1.193 --add-host empresa.asir:15.236.112.115 gerardojuanasir/proyecto_1:projeto-dns-correo
Unable to find image 'gerardojuanasir/proyecto_1:projeto-dns-correo' locally
3_dns-correo: Pulling from gerardojuanasir/proyecto_1
d1669123f281: Already exists
10.98.1.193: Pulling from dockerhub
2a01d17759d0: Pull complete
4477e110e006: Pull complete
cbc9246d99f: Pull complete
25b8951197a1: Pull complete
d239208c9476: Pull complete
d032408c9476: Pull complete
d1034da57701: Pull complete
61bae87e672c: Pull complete
a71092b28755: Pull complete
811033330000: Pull complete
53nnah8eb65f: Pull complete
872a885abddbf: Pull complete
9458c9910e0ec: Pull complete
617574102408: Pull complete
e0f099308540: Pull complete
2a089c3b8c5e: Pull complete
2b77dc42308a: Pull complete
f4f36d5cf23f: Pull complete
b075333d0c434: Pull complete
895403330000: Pull complete
3a920d566e08: Pull complete
Digest: sha256:42be59656ab458270d21d71ae6f8eadede9dc86424ad1d9b5fe99db0d56349
Status: Downloaded newer image for gerardojuanasir/proyecto_1:projeto-dns-correo
72e76697a97: Pull complete
72e76697a97: Pull complete
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
72e76697a97 gerardojuanasir/proyecto_1:projeto-dns-correo "/entrypoint.sh" 5 seconds ago Up 5 seconds 3_dns-correo
bla333c4250f gerardojuanasir/proyecto_1:projeto-ftp "/entrypoint.sh" 7 minutes ago Up 7 minutes 2_ftp
38876742028f gerardojuanasir/proyecto_1:projeto-http "apache2ctl -D FOREGROUND" 12 minutes ago Up 12 minutes 1_http
root@ip-10-98-1-193:/home/ubuntu# docker ps
```

Una vez lanzado, comprobamos que los tres servidores corren a la vez sin problema.

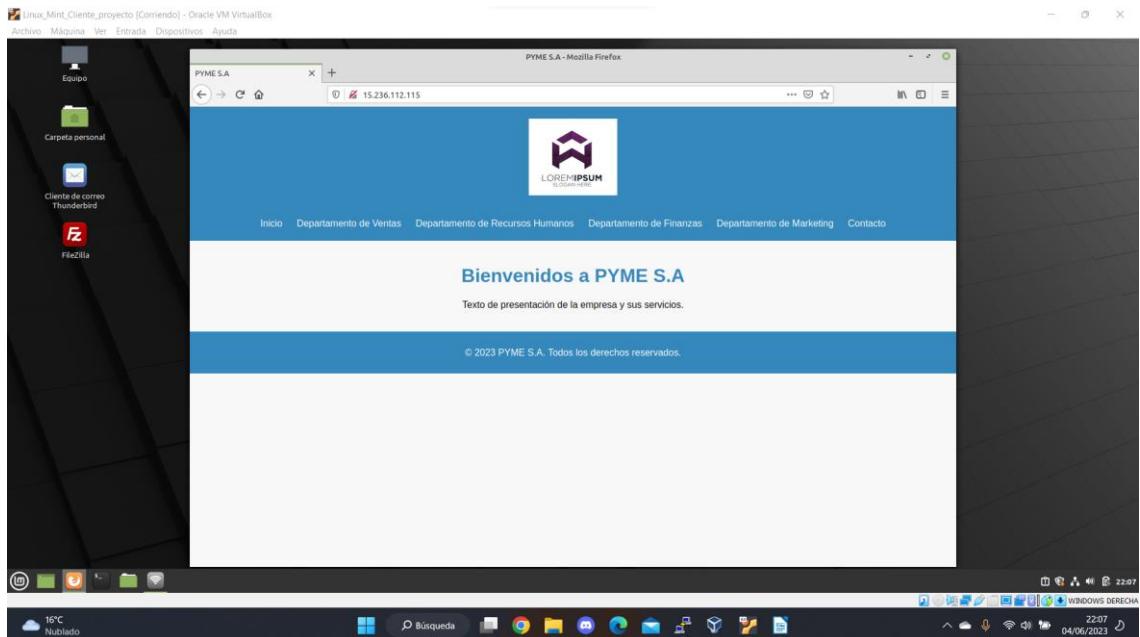
Y así termina la parte de administrador del proyecto. Sólo queda el manual de usuario, que vendrá a continuación. Pulsa [aquí](#) para volver arriba.

## 10.12 Anexo 12: Manual de usuario/prueba de funcionamiento.

Un usuario de la empresa que implemente nuestro sistema deberá tener unos conocimientos mediosbajos de informática. Con manejo de navegador de Internet, Filezilla, Thunderbird y saber cambiar el servidor DNS le bastaría.

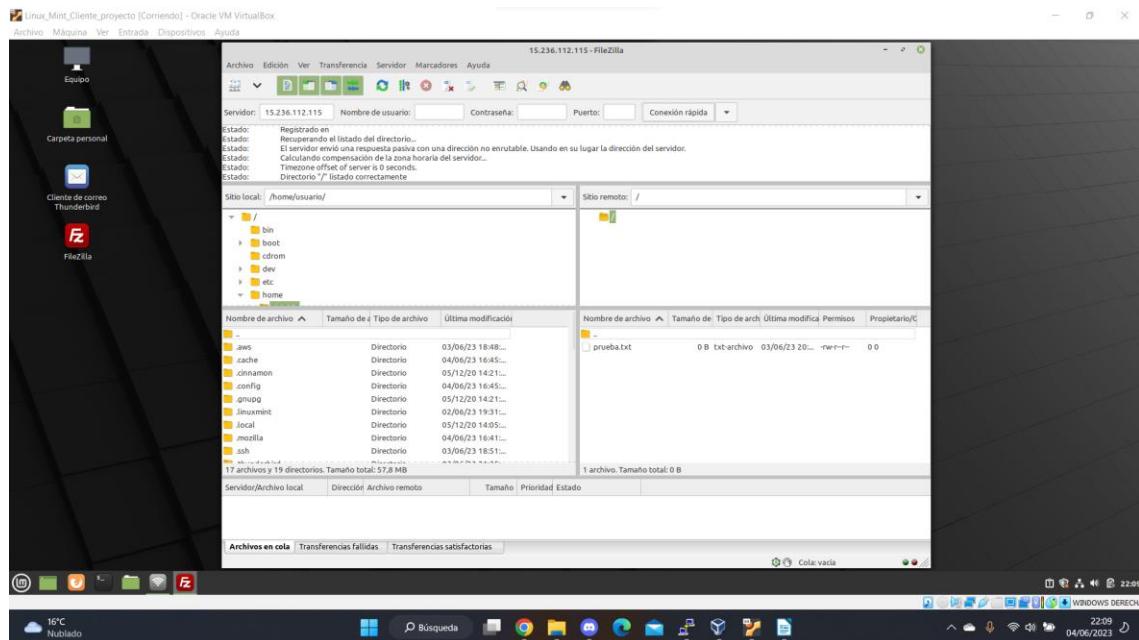
Si quisiera acceder a la web de la empresa, deberá abrir el navegador e introducir la siguiente dirección “<http://15.236.112.115>”. La IP corresponde a la IP pública del bastión, que se encargará de reenviar la petición al servidor.

Como podemos observar, el servidor responde a la perfección y se muestra la web.

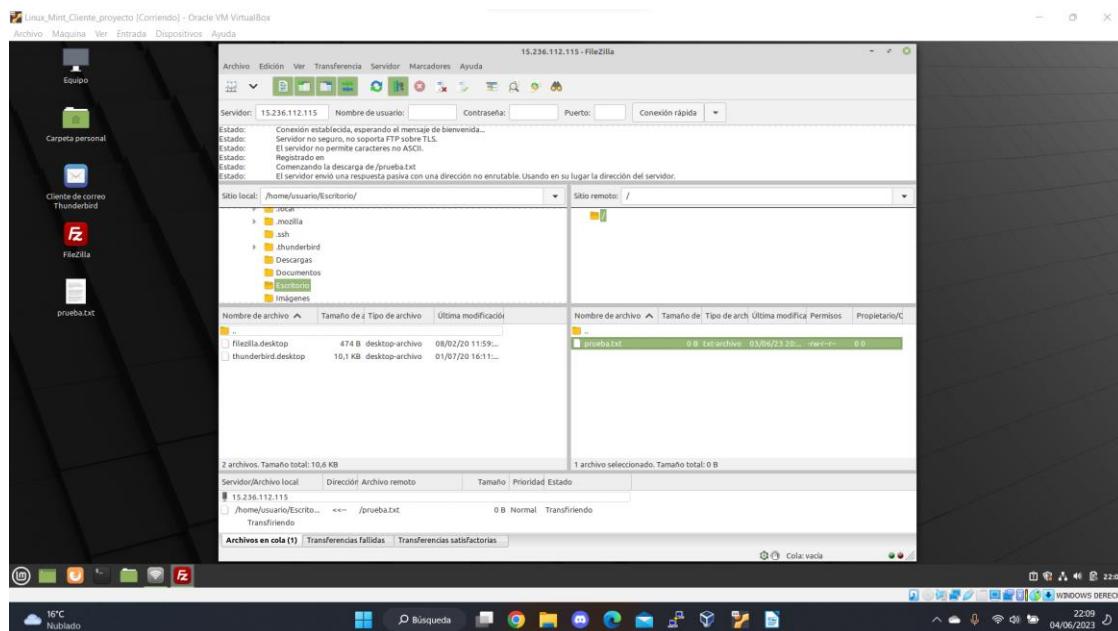


## Dockerización de un servidor completo en Cloud.

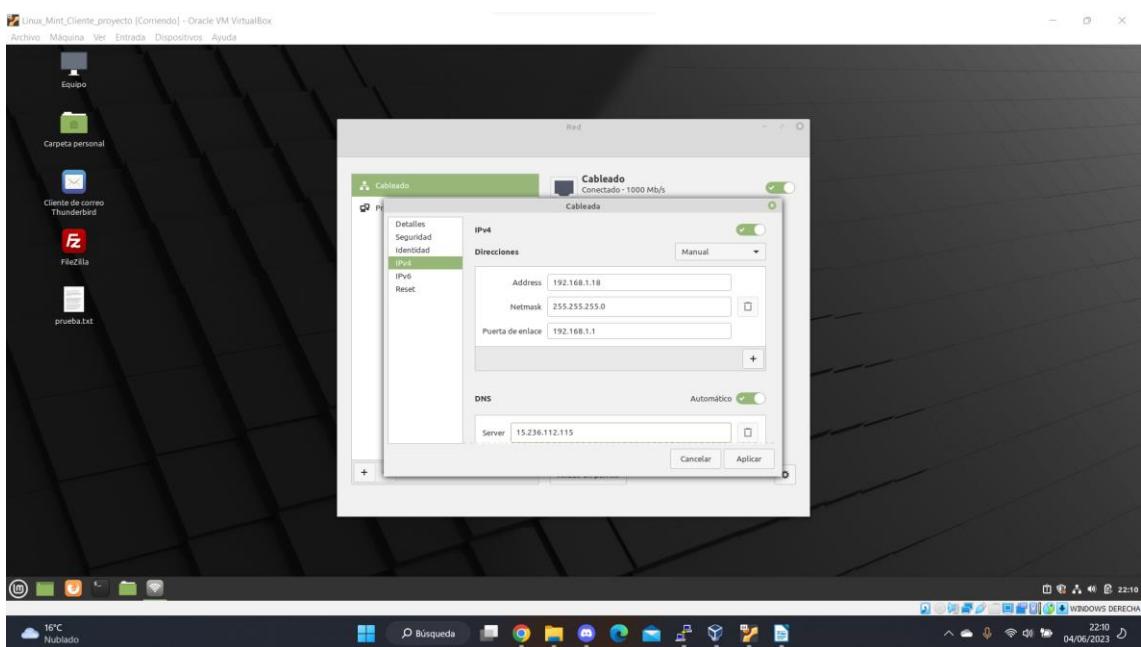
Para acceder a los archivos almacenados en el servidor FTP por su empresa, deberá abrir Filezilla y colocar la misma IP en la caja de texto ‘Servidor’. Sin ningún dato más, damos a conexión rápida, aceptamos la ventana que se nos abre y estará establecida la conexión, en la que visualizamos el fichero de prueba.



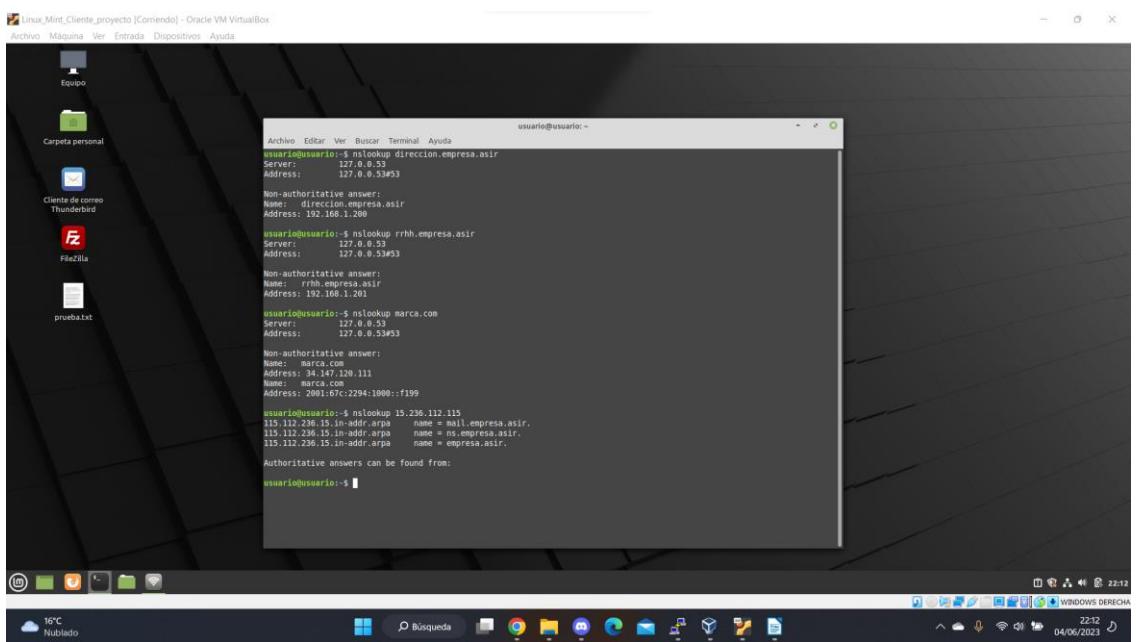
Si probamos a bajarnos el fichero, vemos que podemos hacerlo sin problema.



Para probar el servidor DNS, el usuario tendría primero que colocar la IP pública del bastión como servidor DNS. Para ello, tendrá que abrir la configuración de red IPv4 y escribir la dirección en ‘Server’, en el apartado DNS.



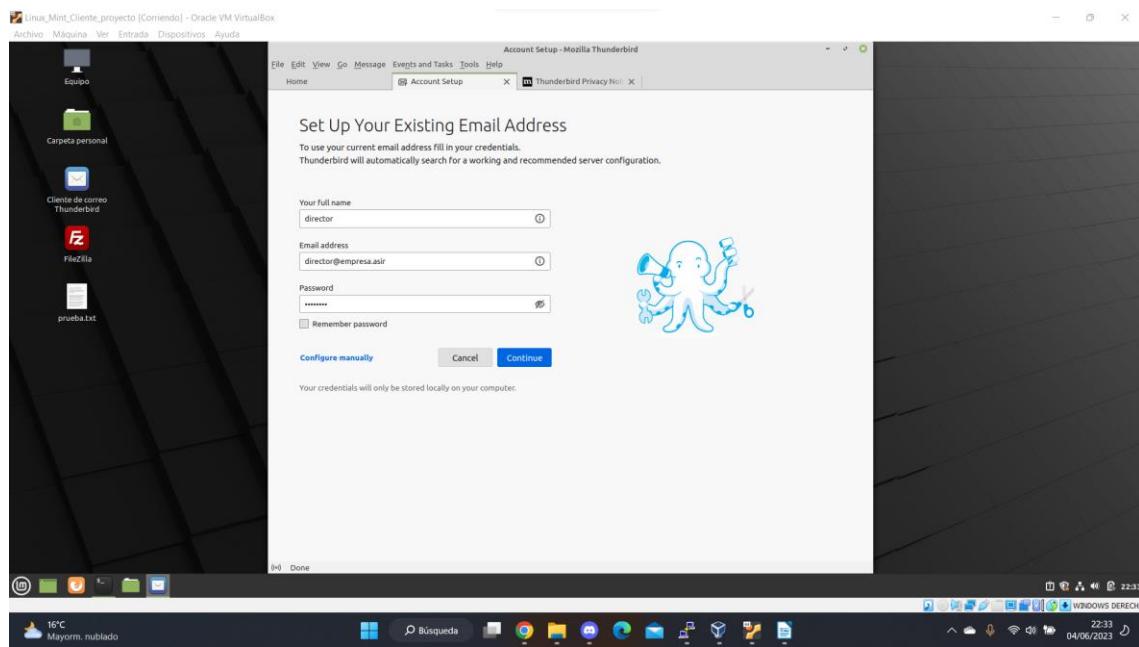
Para ver una resolución de nombres, podría ejecutar el comando ‘nslookup’. Como vemos en la captura, se resuelven tanto nombres de dominio internos como externos.



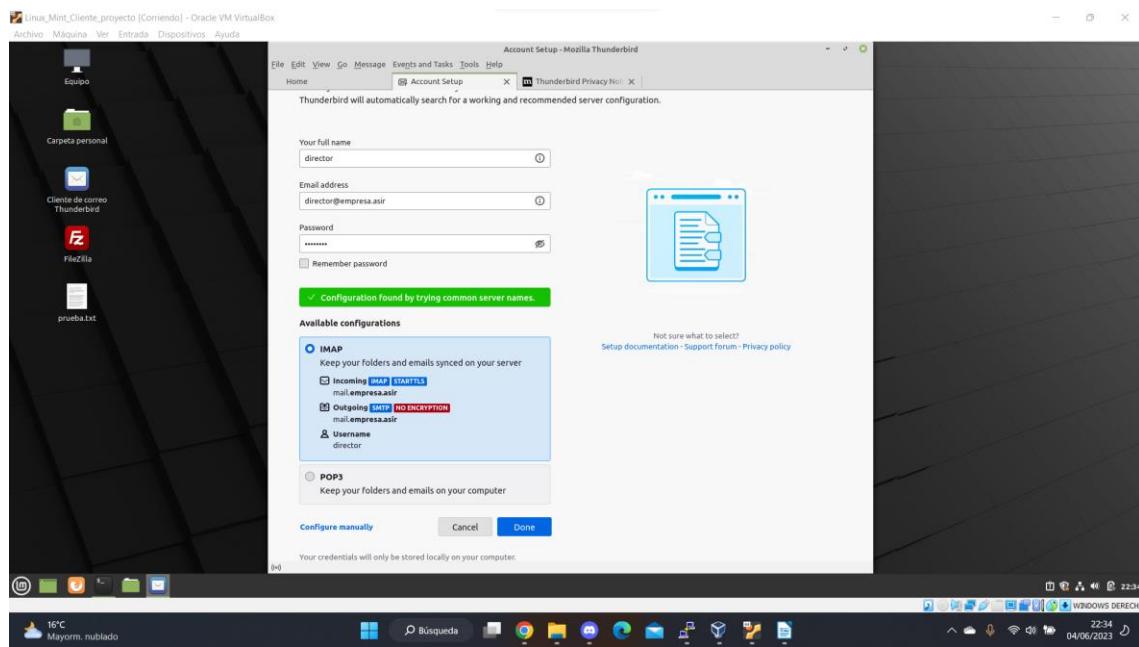
También podría probarlo accediendo a una página web externa a través del navegador, ya que el servidor DNS se encargaría de traducir a su correspondiente IP la dirección que introduzca el usuario.

## Dockerización de un servidor completo en Cloud.

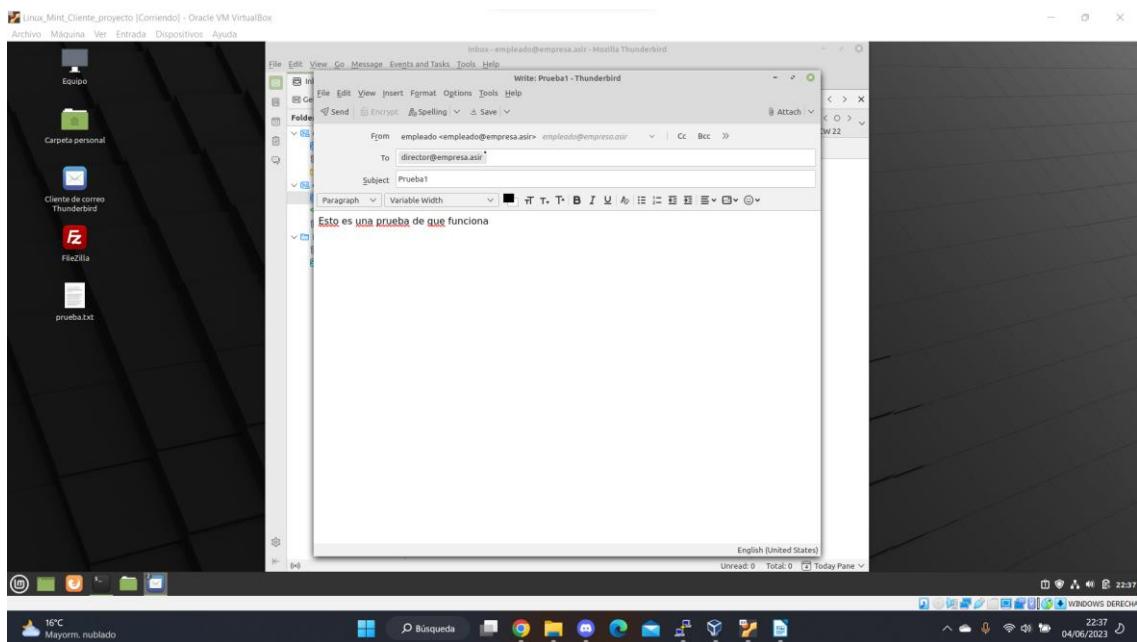
Para usar el correo electrónico, el usuario tendría a su disposición la aplicación Thunderbird. Primero, tendría que hacer login con su nombre de usuario, su dirección de correo y su contraseña. Para la demostración, vamos a utilizar los usuarios ‘director’ y empleado. Una vez introducidos, le daría a ‘Continuar’.



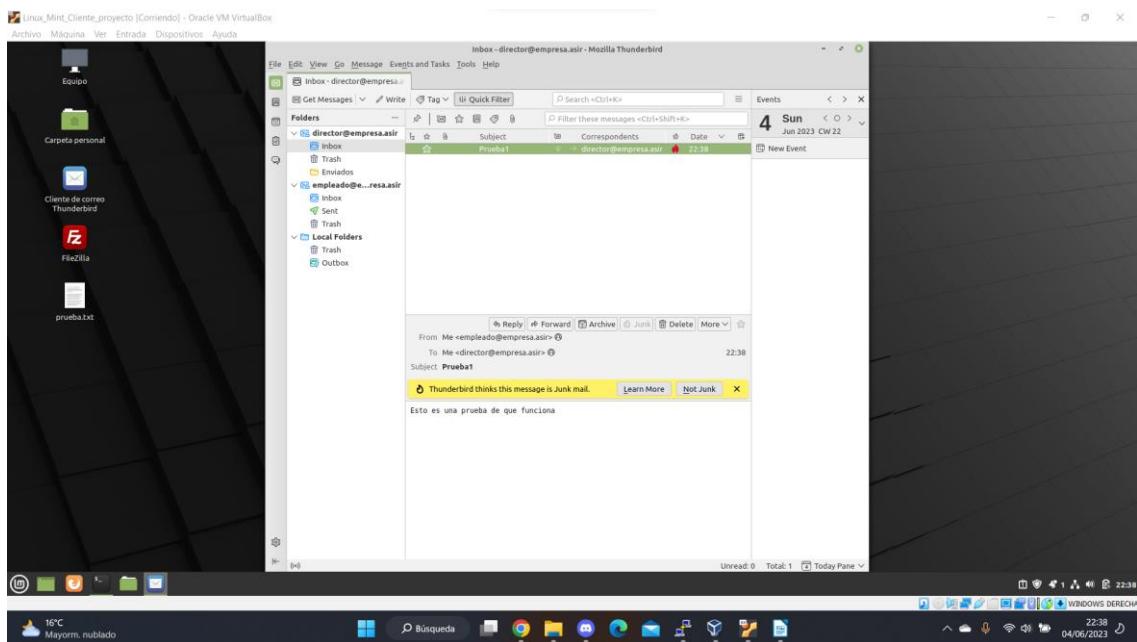
Thunderbird se encargaría de completar la configuración de protocolos y puertos. El usuario sólo tendría que darle a ‘Continue’.



Una vez en la pantalla principal de Thunderbird, el usuario tendrá que darle a ‘Write’ para escribir un mail. Se abrirá una ventana en la que poder hacerlo. Pulsará en ‘Send’ para enviarlo.



Como vemos, el destinatario del correo lo recibe de inmediato, pudiendo leerlo sin ningún problema.



Hasta aquí la prueba de usuario/funcionamiento. Pulsa [aquí](#) para volver arriba.

## 10.13 Anexo 13: Manual de registro en Git Hub y subida de proyecto.

Pulsa [aquí](#) para volver arriba.