

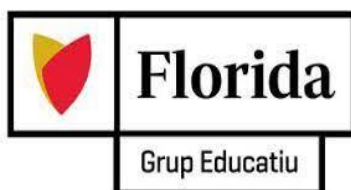


GENERALITAT
VALENCIANA

Conselleria d'Educació,
Cultura i Esport

DOCKER FACILIDAD

RUBEN QUINTANILLA ALMENAR



Índice

1. Resumen del proyecto:	3
2. Justificación y objetivos del proyecto:	4
3. Desarrollo del proyecto	5
3.1 - <i>Análisis de mercado:</i>	5
3.1.1 - <i>Método de explotación</i>	6
3.1.2 - <i>Modelo de negocio</i>	6
3.2 - <i>Metodologías:</i>	7
3.3 - <i>Descripción de los componentes:</i>	8
3.4 - <i>Problemas/dificultades:</i>	9
4. Comandos:	11
4.1 - <i>Instalación de Docker</i>	11
4.2- <i>Creación de la imagen</i>	12
4.3- <i>Repositorio git hub</i>	13
4.4 - <i>Crear el script de copia de seguridad:</i>	14
4.4.1 - <i>Mejora de script-copia Incrementada</i>	15
4.5 – <i>DockerFile</i>	16
4.6 - <i>Build</i>	17
4.7 – <i>Run</i>	17
5. Líneas futuras de trabajo	18
6.Conclusión	19
7.Webgrafia	20

Introducción

En el presente trabajo, se presenta el proceso de creación de una copia de seguridad automatizada utilizando Docker y Git hub. La copia de seguridad se realiza en una aplicación de muestra, que puede ser una aplicación web o una base de datos simple. El objetivo del trabajo es demostrar cómo se puede automatizar el proceso de copia de seguridad de una aplicación, lo que permitirá al usuario crear diferentes versiones de la copia de seguridad y restaurar la aplicación a una versión anterior si es necesario.

Para llevar a cabo el proceso de copia de seguridad, se utilizan diferentes tecnologías y recursos, como Docker y composición de Docker, Git un servicio de repositorio, un servicio de almacenamiento en la nube (opcional), un lenguaje de secuencias de comandos como Bash, y herramientas como Tar o Rsync para crear las copias de seguridad.

El proceso de creación de una copia de seguridad automatizada es fundamental para garantizar la integridad y seguridad de los datos de una aplicación, especialmente en entornos empresariales donde los datos son cruciales para el funcionamiento de la organización. Por lo tanto, este trabajo es de gran importancia para cualquier persona que busque implementar una solución de copia de seguridad automatizada en su aplicación.

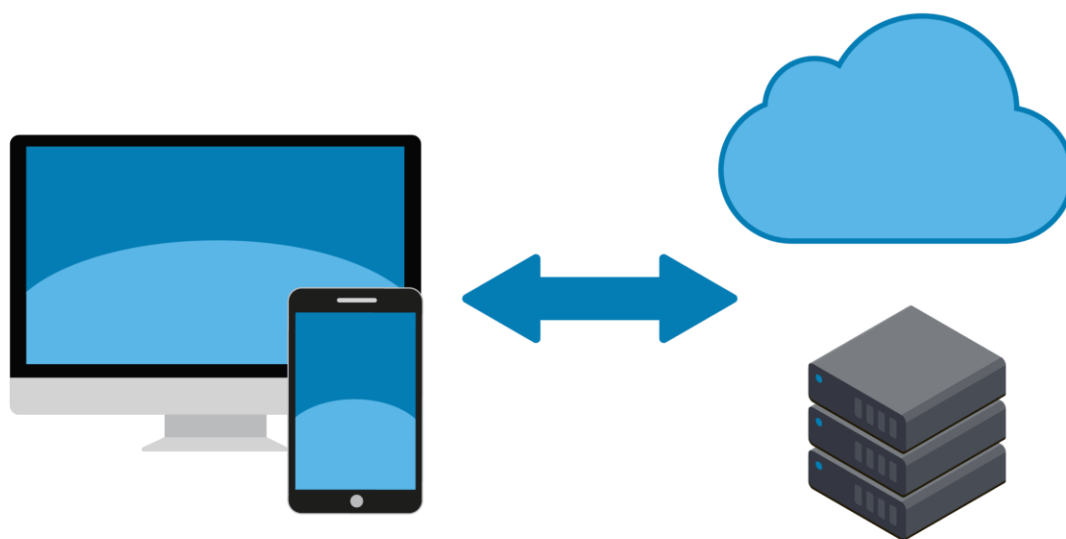
1. Resumen del proyecto:

El proyecto consiste en la creación de un contenedor Docker que se encarga de realizar copias de seguridad de aplicaciones y subirlas a un repositorio en GitHub. El contenedor incluye un script que utiliza el comando tar para crear un archivo comprimido con la copia de seguridad y luego utiliza Git para subir ese archivo al repositorio de GitHub.

El contenedor está formado por una imagen de Apache que se utiliza como base y se instala Git para permitir la subida de archivos al repositorio. Además, se incluye un script de shell que realiza la copia de seguridad y lo configura como una tarea de cron que se ejecuta diariamente a las 11 a.m.

El proyecto tiene como objetivo proporcionar una solución automatizada y fácil de usar para realizar copias de seguridad de aplicaciones y almacenarlas en un lugar seguro. Además, el uso de contenedores Docker permite una fácil portabilidad y escalabilidad del sistema en diferentes entornos.

Ilustración 1-1



2. Justificación y objetivos del proyecto:

El proyecto se propone como una solución para la gestión y la automatización de copias de seguridad de aplicaciones y bases de datos utilizando tecnologías de virtualización y control de versiones.

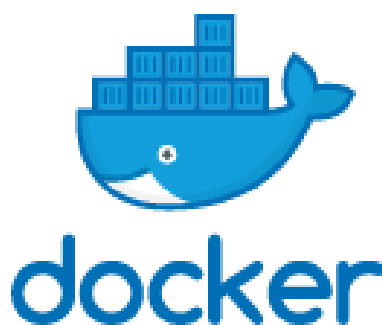
Este proyecto está dirigido a desarrolladores de aplicaciones y administradores de sistemas que buscan una solución escalable, confiable y fácil de usar para realizar copias de seguridad de sus aplicaciones y datos.

Los objetivos del proyecto son los siguientes:

- Proporcionar una solución fácil y escalable para realizar copias de seguridad de aplicaciones y bases de datos en entornos de contenedores Docker.
- Automatizar el proceso de copia de seguridad utilizando scripts para minimizar la intervención humana.
- Utilizar Git para el control de versiones de las copias de seguridad, lo que permite una fácil reversión a versiones anteriores en caso de problemas o errores.
- Permitir el almacenamiento de las copias de seguridad en la nube utilizando servicios de almacenamiento en la nube como Amazon S3 o Google Cloud Storage.
- Proporcionar una documentación clara y detallada para que los usuarios puedan comprender y utilizar fácilmente la solución.

En general, el objetivo final del proyecto es proporcionar una solución confiable y escalable para la gestión y automatización de copias de seguridad de aplicaciones y bases de datos, que permita a los usuarios proteger sus datos críticos de forma segura y fácil de usar.

Ilustración 1-2



3. Desarrollo del proyecto

3.1 - Análisis de mercado:

Como modelo de negocio, el uso de Docker y contenedores se ha vuelto cada vez más popular en los últimos años. La capacidad de Docker para simplificar el desarrollo, implementación y administración de aplicaciones en contenedores ha llevado a una amplia adopción por parte de empresas y organizaciones de todos los tamaños.

Puntos importantes:

1. Identificación del mercado objetivo: el mercado objetivo incluir empresas y organizaciones que deseen simplificar su proceso de implementación y administración de aplicaciones mediante el uso de contenedores.
2. Competencia: el mercado de Docker está muy competido, con varias opciones para elegir, como Kubernetes, Mesos, Amazon ECS, etc. Por lo tanto, es importante diferenciarse de la competencia y ofrecer un valor agregado único.
3. Tendencias del mercado: la tendencia en el mercado de Docker es el aumento de la adopción por parte de empresas y organizaciones de todos los tamaños. También se espera un mayor crecimiento en la implementación de contenedores en la nube, ya que las empresas buscan aumentar la escalabilidad y reducir los costos.
4. Análisis de los ingresos: el modelo de negocio para el proyecto de Docker puede incluir la venta de servicios de soporte, consultoría y capacitación en contenedores, así como la venta de soluciones de software basadas en contenedores. También puede ofrecer servicios de alojamiento y administración de contenedores para los clientes que no deseen administrar su propia infraestructura.

Ilustración 1-3



5. Barreras de entrada: las barreras de entrada para el mercado de Docker son relativamente bajas, ya que es un mercado en crecimiento y hay una gran cantidad de recursos disponibles en línea para ayudar a las personas a aprender y comenzar con Docker. Sin embargo, la competencia es alta y es importante ofrecer un valor único para destacar en el mercado.

En resumen, el mercado de Docker está en constante crecimiento y tiene un gran potencial para los modelos de negocio basados en contenedores. Sin embargo, es importante diferenciarse de la competencia y ofrecer un valor agregado único para destacar en el mercado.

3.1.1 - Método de explotación

En términos de modelo de negocio, una posible forma de explotar esta solución podría ser ofrecer servicios de copias de seguridad automatizadas a clientes que utilicen aplicaciones críticas en la nube. La empresa podría cobrar por el almacenamiento de las copias de seguridad y por la configuración y mantenimiento del sistema de copias de seguridad automatizado.

3.1.2 - Modelo de negocio

Sería ofrecer un servicio de copias de seguridad automatizadas en la nube para aplicaciones web mediante el uso de contenedores Docker. El servicio podría ofrecer diferentes planes de suscripción basados en la cantidad de aplicaciones, el tamaño de los datos a respaldar y la frecuencia de las copias de seguridad. Además, se podrían ofrecer opciones de almacenamiento en diferentes proveedores de nube como Amazon S3, Google Cloud Storage o Microsoft Azure.

Otra opción podría ser ofrecer servicios de consultoría y asesoramiento a empresas que deseen implementar soluciones de copias de seguridad automatizadas en su infraestructura de aplicaciones web, utilizando tecnologías como Docker y Git. Este servicio podría incluir evaluaciones de infraestructura, diseño e implementación de soluciones de copias de seguridad personalizadas, capacitación y soporte técnico continuo

3.2 - Metodologías:

En términos generales, la metodología utilizada en Docker para hacer copias de seguridad y subirlas a GitHub se basa en los siguientes pasos:

1. Identificar los requisitos de la copia de seguridad: es necesario definir qué archivos o datos se deben respaldar, con qué frecuencia, dónde se almacenarán las copias de seguridad y cómo se realizarán las pruebas de recuperación.
2. Crear un script de copia de seguridad: se debe crear un script que ejecute las acciones necesarias para realizar la copia de seguridad, como la creación de un archivo comprimido o la copia de los datos a un servidor remoto.
3. Configurar Git para almacenar las copias de seguridad: se debe crear un repositorio en GitHub o en otro servicio de control de versiones y configurar el entorno de Docker para que pueda acceder a este repositorio.
4. Crear un archivo Dockerfile: se debe crear un archivo Dockerfile que contenga las instrucciones para construir la imagen de Docker que se utilizará para ejecutar el script de copia de seguridad.
5. Construir la imagen de Docker: se debe utilizar el archivo Dockerfile para construir la imagen de Docker que se utilizará para ejecutar el script de copia de seguridad.
6. Ejecutar la imagen de Docker: se debe ejecutar la imagen de Docker para realizar la copia de seguridad y subir los archivos a GitHub.
7. Configurar una tarea programada: se debe configurar una tarea programada en Docker para que la copia de seguridad se realice automáticamente en intervalos regulares.

3.3 - Descripción de los componentes:

Como se mencionó anteriormente, el proyecto consiste en crear una solución de copia de seguridad utilizando Docker y Git. A continuación, se describen los componentes de la aplicación:

- Mockups: No se han diseñado mockups ya que se trata de una solución de infraestructura.
- Arquitectura: La arquitectura de la solución está compuesta por los siguientes elementos:
 - Aplicación de muestra: Se utilizará una aplicación de muestra para realizar una copia de seguridad, que puede ser una aplicación web o una base de datos simple.
 - Docker: Se utilizará Docker para crear una imagen que incluya los scripts y las herramientas necesarias para realizar la copia de seguridad.
 - Git: Se utilizará Git para almacenar las diferentes versiones de las copias de seguridad.
 - Tar y Rsync: Se utilizarán estas herramientas para comprimir las copias de seguridad.
 - Almacenamiento en la nube: Si se desea, se puede utilizar un servicio de almacenamiento en la nube como Amazon S3 o Google Cloud Storage para almacenar las copias de seguridad.
 - Bash: Para configura la infraestructura y crear los scripts necesarios.
- Backend: El backend de la aplicación estará compuesto por un script de Bash automatice el proceso de copia de seguridad, como crear un archivo tar de los datos, y luego use Git hub para confirmar y enviar la copia de seguridad al repositorio.
- Tecnologías utilizadas: Las principales tecnologías utilizadas en la solución son Docker, Git, Tar, Rsync y Bash.



Ilustración 1-4

En resumen, la aplicación consiste en utilizar Docker para crear una imagen que incluya los scripts y herramientas necesarias para realizar la copia de seguridad de una aplicación de muestra, utilizando Git para almacenar las diferentes versiones de las copias de seguridad. Además, se utilizarán herramientas como Tar y Rsync para crear las copias de seguridad y un script de Bash para automatizar el proceso de copia de seguridad.

3.4 - Problemas/dificultades:

Durante el desarrollo del PFC, se encontraron algunas dificultades que afectaron el proceso de implementación y pruebas de la aplicación. Uno de los principales problemas fue la configuración inicial de Docker, que resultó ser más compleja de lo esperado y requirió de una investigación adicional para solucionarlo.

Además, la integración de Git para la gestión de versiones también presentó algunos desafíos, como la resolución de conflictos la gestión de permisos de usuario en los repositorios.

Para solucionar estos problemas, se realizaron diferentes acciones, como la consulta de documentación y foros de soporte técnico para Docker y Git.

En general, estas dificultades y desafíos permitieron mejorar el proceso de desarrollo y aprendizaje en el proyecto, lo que llevó a una mayor comprensión de las tecnologías utilizadas y su aplicación en situaciones reales.



Ilustración 1-5

3.4 - Resultados obtenidos:

En cuanto a los resultados obtenidos en este proyecto, se ha logrado desarrollar una solución de copia de seguridad automatizada para una aplicación de muestra utilizando Docker, Git y las herramientas de tar y rsync. Se ha creado una imagen de Docker que incluye los scripts y herramientas necesarias para realizar la copia de seguridad y se ha configurado un repositorio Git para almacenar las diferentes versiones de las copias de seguridad.

Además, se ha automatizado el proceso de copia de seguridad mediante un script que crea un archivo tar de los datos y utiliza Git para confirmar y enviar la copia de seguridad al repositorio. También se ha demostrado la capacidad de volver a una versión anterior de la copia de seguridad revisando una confirmación anterior en el repositorio de Git y restaurando los datos a la aplicación original.

En general, se ha logrado desarrollar una solución eficiente y escalable para la copia de seguridad de aplicaciones utilizando tecnologías modernas y herramientas de automatización, lo que puede ser útil para empresas y organizaciones que necesiten una solución de copia de seguridad robusta y confiable.

Ilustración 1-6

Ilustración 1-6

 Nuber002	Copia de seguridad del 2023-04-28	2c967b4 6 minutos ago	 6 commits
 backup-2023-04-28_12-33-01.tar.gz	Copia de seguridad del 2023-04-28	11 minutos ago	
 backup-2023-04-28_12-34-01.tar.gz	Copia de seguridad del 2023-04-28	10 minutos ago	
 backup-2023-04-28_12-35-01.tar.gz	Copia de seguridad del 2023-04-28	9 minutos ago	
 backup-2023-04-28_12-36-01.tar.gz	Copia de seguridad del 2023-04-28	8 minutos ago	
 backup-2023-04-28_12-37-01.tar.gz	Copia de seguridad del 2023-04-28	7 minutos ago	
 backup-2023-04-28_12-38-01.tar.gz	Copia de seguridad del 2023-04-28	6 minutos ago	
 backup.snar	Copia de seguridad del 2023-04-28	6 minutos ago	
 copia	Copia de seguridad del 2023-04-28	11 minutos ago	

4. Comandos:

4.1 - Instalación de Docker

- Actualización de los paquetes:

```
sudo apt-get update.
```

- Permitir la descarga de paquetes a través de HTTPS:

```
sudo apt-get install apt-transport-https ca-certificates curl  
gnupg-agent software-properties-common.
```

- Agregar la clave GPG de Docker:

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

- Agregar el repositorio de Docker a sus fuentes de APT:

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

- Actualice su lista de paquetes existentes con los nuevos paquetes de Docker:

```
sudo apt-get update
```

- Se va a instalar desde el repositorio de Docker en lugar del repositorio de Ubuntu predeterminado:

```
sudo apt-cache policy docker-ce
```

- Finalmente, instalar Docker:

```
sudo apt-get install docker-ce
```

- Verificar que Docker se instaló correctamente:

```
sudo docker run hello-world
```

4.2-Creación de la imagen

- Crear un contenedor de Docker con una imagen de Apache y configurar la aplicación web. Para ello, se puede utilizar el siguiente comando:

```
root@debian11:/home/pruebas/myapp# docker run -d --name mi-apache httpd:latest
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
26c5c85e47da: Pull complete
2d29d3837df5: Pull complete
2483414a5e59: Pull complete
e78016c4ba87: Pull complete
757908175415: Pull complete
Digest: sha256:al82ef2350699f04b8f8e736747104eb273e255e818cd55b6d7aa50a1490ed0c
Status: Downloaded newer image for httpd:latest
f7b8647ccdb12bfc7a7827ce477a812434620212836fc634687f40707a6d8a15
```

Ilustración 1-7

Este comando crea un contenedor de Docker con la imagen httpd de Apache.

4.3-Repositorio git hub

Para configurar un repositorio Git, primero debe crear una cuenta en un servicio de alojamiento de Git como GitHub o GitLab. Luego, cree un nuevo repositorio y siga las instrucciones para clonar el repositorio en su máquina local.

- Cree una cuenta en GitHub y luego inicie sesión.
- Haga clic en el botón "Nuevo repositorio" en la página de inicio.
- Ingrese un nombre y una descripción para el repositorio y seleccione la opción "Inicializar este repositorio con un archivo README".
- Haga clic en el botón "Crear repositorio".
- En la página del repositorio, haga clic en el botón "Clonar o descargar" y copie la URL del repositorio.

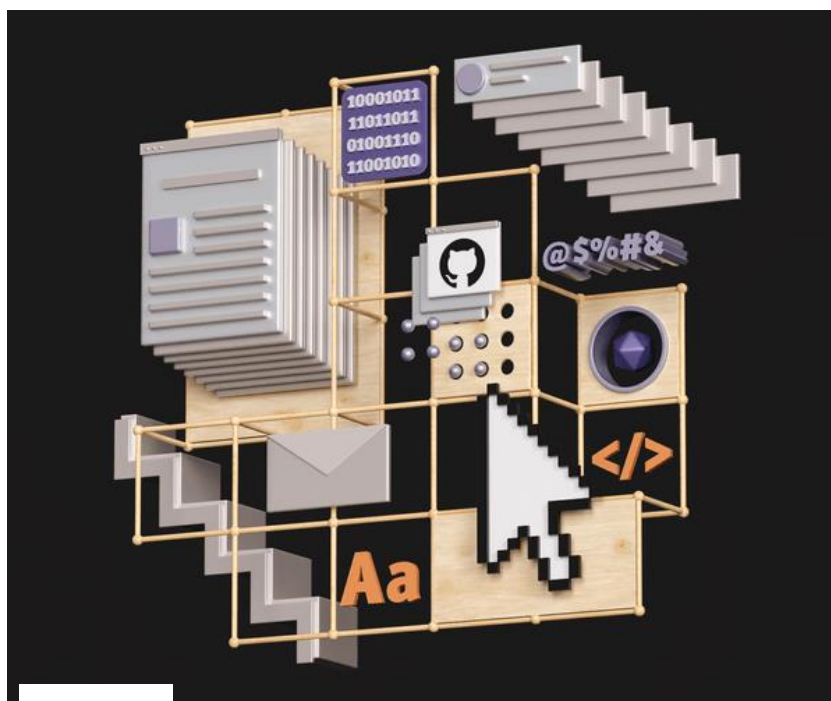


Ilustración 1-8

4.4 - Crear el script de copia de seguridad:

Esta imagen muestra cómo se debería hacer el script de copia de seguridad.

```
#!/bin/bash

# Variables de configuración
BACKUP_DIR=/backup
GIT_REPO="https://ghp_uHuS7sf1qqaHxU9OxFLxydoYb2S5i0E59Jy@github.com/Nuber002/copias-de-seguridad.git"

# Crea un archivo tar con la copia de seguridad
tar -czvf $BACKUP_DIR/backup-$(date +%Y-%m-%d).tar.gz /var/www/html

# Añade y confirma los cambios en Git
cd $BACKUP_DIR
git init
git add .
git config --global user.email "rubenalmemar22@gmail.com"
git commit -m "Copia de seguridad del $(date +%Y-%m-%d)"
git remote remove origin
git remote add origin $GIT_REPO
git push --force origin master
git push -u origin master
```

Ilustración 1-9

- Define dos variables de configuración: `BACKUP_DIR` y `GIT_REPO` que indican el directorio de destino de la copia de seguridad y la dirección del repositorio Git donde se almacenarán los archivos de la copia de seguridad. Generaremos un token en Git hub para poder mandar el comprimido en el git hub.
- Crea un archivo tar con la copia de seguridad utilizando el comando `tar -czvf $BACKUP_DIR/backup-$(date +%Y-%m-%d).tar.gz /var/www/html`. Este comando crea un archivo tar comprimido en formato gzip que contiene los archivos en el directorio `/var/www/html`.
- Estas líneas se encargan de añadir y confirmar los cambios en el repositorio Git. Primero se cambia al directorio donde se encuentran las copias de seguridad y se inicializa el repositorio Git con `git init`. Luego se añaden todos los archivos con `git add`. A continuación se configura el correo electrónico para el usuario con `git config --global user.email`. Después se realiza el commit con un mensaje descriptivo que incluye la fecha de creación del archivo de copia de seguridad. A continuación, se elimina el origen actual y se agrega el nuevo origen, es decir, la URL del repositorio Git. Por último, se fuerza la subida de los cambios con `git push --force` y se establece la relación de seguimiento con `git push -u origin master`.

4.4.1 - Mejora de script-copia Incrementada

Este script es una versión mejorada del anterior, que implementa una copia de seguridad incremental. En lugar de crear una copia de seguridad completa cada vez, la copia de seguridad incremental solo respalda los cambios realizados desde la última copia de seguridad.

```
#!/bin/bash

# Variables de configuración
BACKUP_DIR=/backup
GIT_REPO="https://ghp_uHuS7sflqghaHxU9OxFLxydoYb2S5i0E59Jy@github.com/Nuber002/copias-de-seguridad.git"
SNAR_FILE=$BACKUP_DIR/backup.snar

# Crea un archivo tar con la copia de seguridad incremental
tar -czvf $BACKUP_DIR/backup-$(date +%Y-%m-%d).tar.gz --listed-incremental=$SNAR_FILE /var/www/html

# Añade y confirma los cambios en Git
cd $BACKUP_DIR
git init
git add .
git config --global user.email "rubenalmenar22@gmail.com"
git commit -m "Copia de seguridad del $(date +%Y-%m-%d)"
git remote remove origin
git remote add origin $GIT_REPO
git push --force origin master
git push -u origin master
```

Ilustración 1-10

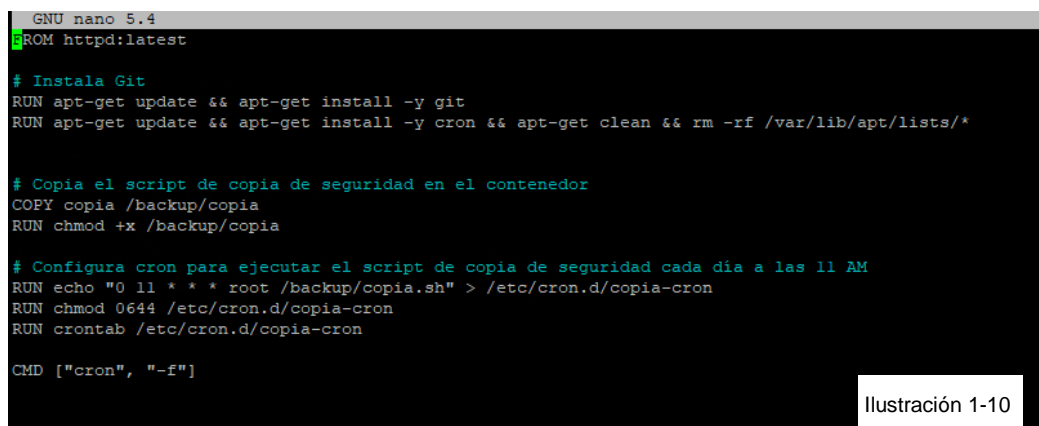
La variable SNAR_FILE almacena la ruta donde se guarda el archivo. snar, que contiene la información de la última copia de seguridad incremental realizada. Esto es necesario para que la próxima vez que se ejecute el script, se pueda comparar la información del archivo. snar con los archivos actuales y determinar cuáles han sido modificados o eliminados desde la última copia de seguridad.

La opción --listed-incremental de la herramienta tar indica la ruta del archivo. snar y le indica a tar que debe crear una copia de seguridad incremental.

Luego de crear la copia de seguridad incremental, se añaden y confirman los cambios en Git, como se hacía anteriormente.

4.5 – DockerFile

Crear el Dockerfile para construir la imagen del contenedor que ejecutará el script de copia de seguridad.



```
GNU nano 5.4
FROM httpd:latest

# Instala Git
RUN apt-get update && apt-get install -y git
RUN apt-get update && apt-get install -y cron && apt-get clean && rm -rf /var/lib/apt/lists/*

# Copia el script de copia de seguridad en el contenedor
COPY copia /backup/copia
RUN chmod +x /backup/copia

# Configura cron para ejecutar el script de copia de seguridad cada día a las 11 AM
RUN echo "0 11 * * * root /backup/copia.sh" > /etc/cron.d/copia-cron
RUN chmod 0644 /etc/cron.d/copia-cron
RUN crontab /etc/cron.d/copia-cron

CMD ["cron", "-f"]
```

Ilustración 1-10

- Se utiliza la imagen base httpd:latest.
- Se instala Git en el contenedor utilizando el comando `RUN apt-get update && apt-get install -y git`.
- Se copia el script de copia de seguridad en el directorio `/backup` dentro del contenedor utilizando el comando `COPY backup.sh /backup/backup.sh`.
- Se asignan permisos de ejecución al script de copia de seguridad utilizando el comando `RUN chmod +x /backup/backup.sh`.
- Se configura cron para que ejecute el script de copia de seguridad cada día a las 11 AM utilizando los comandos `RUN echo "0 11 * * * root /backup/backup.sh" > /etc/cron.d/backup-cron`, `RUN chmod 0644 /etc/cron.d/backup-cron` y `RUN crontab /etc/cron.d/backup-cron`.
- Finalmente, se establece el comando por defecto del contenedor como `CMD ["cron", "-f"]`, lo que indica que debe ejecutarse el servicio cron en el contenedor para que el script de copia de seguridad se ejecute automáticamente.

4.6 - Build

Construir la imagen del contenedor utilizando el siguiente comando

```
root@debian11:/home/backup# docker build -t backup-container /home/backup/
[+] Building 16.0s (13/13) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 571B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/httpd:latest
=> [1/8] FROM docker.io/library/httpd:latest
=> [internal] load build context
=> => transferring context: 27B
=> CACHED [2/8] RUN apt-get update && apt-get install -y git
=> [3/8] RUN apt-get update && apt-get install -y cron && apt-get clean && rm -rf /var/lib/apt/lists/*
=> [4/8] COPY copia /backup/copia
=> [5/8] RUN chmod +x /backup/copia
=> [6/8] RUN echo "0 11 * * * root /backup/copia.sh" > /etc/cron.d/copia-cron
=> [7/8] RUN chmod 0644 /etc/cron.d/copia-cron
=> [8/8] RUN crontab /etc/cron.d/copia-cron
=> exporting to image
=> => exporting layers
=> => writing image sha256:ca6c611245bfe8d67cebba610d1a32dc54388938bf617f52ed013b16922c1f5c
=> => naming to docker.io/library/backup-container
```

Ilustración 1-11

Para construir la imagen del contenedor, se utiliza el comando `docker build -t backup-container`. Este comando construye la imagen del contenedor utilizando el archivo Dockerfile que se ha creado en el directorio actual (.) y la etiqueta `-t backup-container` asigna un nombre a la imagen.

4.7 – Run

Ejecutar el contenedor utilizando el siguiente comando:

```
root@debian11:/home/backup# docker run -d --name backup backup-container
533fee47fcc5a498462da57e857ab3fada2444f27189ecb04028dac110dc69f4
root@debian11:/home/backup# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
533fee47fcc5	backup-container	"cron -f"	About a minute ago	Up About a minute	80/tcp	backup
f7b8647ccdb1	httpd:latest	"httpd-foreground"	About an hour ago	Up About an hour	80/tcp	mi-apache

Ilustración 1-12

Para ejecutar el contenedor, se utiliza el comando `docker run -d --name backup backup-container`. Este comando ejecuta el contenedor en segundo plano (-d), asigna un nombre al contenedor (--name backup) y utiliza la imagen del contenedor que se ha creado previamente (backup-container). El contenedor se ejecutará automáticamente cada día a las 11 AM y realizará una copia de seguridad de los archivos en el directorio `/var/www/html` y los subirá a un repositorio Git.

5. Líneas futuras de trabajo

Puntos que se puede mejorar el proyecto:

- Implementación de un mecanismo de alerta: se podría agregar un sistema de alerta que notifique al administrador del sistema cuando la copia de seguridad ha fallado o si se han encontrado errores.
- Integración con proveedores de almacenamiento en la nube: se podría agregar la opción de hacer copias de seguridad en servicios en la nube, como Amazon S3 o Google Cloud Storage, para una mayor flexibilidad y redundancia.
- Mejora de la interfaz de usuario: se podría mejorar la interfaz de usuario de la aplicación web, agregando más opciones y características, para que sea más fácil de usar y personalizar.
- Añadir más herramientas y opciones de personalización: se podría agregar más herramientas de automatización y opciones de personalización, como la elección de cuántas copias de seguridad mantener y cuántas eliminar.
- Implementación de la autenticación de usuarios: se podría agregar la autenticación de usuarios y la gestión de permisos para que solo los usuarios autorizados puedan hacer copias de seguridad o restaurar los datos.

6.Conclusión

La implementación de Docker y Git en la realización de copias de seguridad automatizadas ofrece una solución efectiva y eficiente para la gestión de versiones y la recuperación de datos. Utilizando un contenedor de Docker y scripts personalizados que integran herramientas como tar y rsync, es posible realizar copias de seguridad regulares y enviarlas a un repositorio de Git para un almacenamiento seguro. Además, la capacidad de volver a una versión anterior proporciona una forma sencilla de recuperar datos perdidos o dañados. En resumen, la combinación de Docker y Git es una solución ideal para la gestión de copias de seguridad y la recuperación de datos en cualquier aplicación o entorno.

7.Webgrafia

1. Documentación de GIT HUB. Disponible en [<https://docs.github.com/es>].
2. Documentación de DOCKER. Disponible en [<https://docs.docker.com/>].
3. Documentación de Automatización de scripts. Disponible en [<https://www.redeszone.net/tutoriales/servidores/cron-crontab-linux-programar-tareas/>].