

## **Instalacion y configuracion de Docker en GNU/Linux Fedora 36**

### **¿Que es Docker?**

Docker es una plataforma o ecosistema enfocado en la creación y ejecución de contenedores.

### **Ecosistema Docker**

Contenedor(Container) : Unidad estandarizada de software que permite a los desarrolladores aislar desu entorno una aplicación.

Docker Imagen: Una imagen es un archivo con todas las dependencias y configuraciones requeridaspara ejecutar un programa.

Docker File: Es un archivo o un documento de texto simple que incluye una seria de instrucciones quese necesitan ejecutar de manera consecutiva procesos dentro de una imagen.

### **Ventajas de Docker:**

1. Separa los ambientes de desarrollo en sus diferentes versiones
2. Organiza en ambientes de desarrollo separados las diferentes aplicaciones de uso encontenedores.
3. Al desplegar un contenedor consume recursos como un proceso y no como una maquinavirtual.
4. Los contenedores permiten aislar y escalar servicios fácilmente según lo que se necesite.

### **Comandos Básicos de Docker:**

>Docker version  
>Docker info  
>Docker run  
>Docker ps  
>Docker ps -a  
>Docker exec  
>Docker image  
>Docker rm  
>Docker compose  
>Docker start  
>Docker stop  
>Docker volume  
>Docker logs  
>Docker network  
>Docker pull

## Instalación y configuración de Docker e imágenes de nodos LN

Removemos cualquier version anterior de Docker que tengamos instalados en nuestro SO.

```
>sudo dnf remove -y docker docker-client docker-client-latest docker-common docker-latest  
docker-latest-logrotate docker-logrotate docker-selinux docker-engine-selinux docker-engine
```

Adjuntamos el repositorio de Docker .

```
>sudo dnf -y install dnf-plugins-core
```

```
>sudo dnf config-manager --add-repo https://download.docker.com/linux/fedora/docker-ce.repo
```

Instalando Docker Engine.

```
>sudo dnf install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

### Inicializar el Docker Service

```
>sudo systemctl enable --now docker
```

Revisando la version instalada de Docker

```
>docker -v
```

Testeando la version instalada de Docker y corriendo un contenedor hola-mundo

```
>sudo docker run -it fedora echo Hello-World
```

### Permitir a los usuarios no root ejecutar comandos Docker.

Por lo tanto, para permitir que los usuarios estándar de Linux ejecuten los comandos de Docker, añadiendo el usuario al grupo Docker.

```
>sudo usermod -aG docker w2k31984
```

A continuación, cierre la sesión y vuelva a iniciarla y ejecute los comandos de Docker sin el prefijo sudo en el terminal.

```
>docker run hello-world
```

Validando de nuevo información extendida de version de docker

```
>docker version
```

## Instalando Docker Desktop en GNU/Linux Fedora 36

1. Descargamos el paquete de Docker Desktop desde el repositorio con la instrucción siguiente:

```
>wget https://desktop.docker.com/linux/main/amd64/docker-  
desktop-4.11.1-x86_64.rpm
```

2. Ejecutaremos el siguiente comando para instalar Docker Desktop:

```
>sudo dnf install ./docker-desktop-4.11.1-x86_64.rpm
```

Validando con interfaz de linea de comandos.

```
>docker version
```

```
>docker ps
```

```
>docker ps -a    Ver los contenedores activos.
```

```
>docker run -d -p 80:80 docker/getting-started ---Contenedor de  
prueba
```

```
>docker image ls  Listado de imagenes.
```

### Agregando parametros a Docker para descargar imagen de Distro

```
>docker run -d -it ubuntu /bin/bash
```

```
>docker ps
```

#### Levantando el contenedor y servicios de Ubuntu

```
>docker exec -it 08c86d20b5bc /bin/bash (Container ID )
```

Esta instrucción los llevara dentro del contenedor y los archivos.

```
# ls
```

#### Creando archivos dentro del contenedor de Ubuntu.

```
>docker exec -it 08c86d20b5bc touch /tmp/libreriadelsatochi.txt
```

#### Consultando el archivo creado en el directorio.

```
>cd tmp/
```

```
>ls
```

#### Para salir de este directorio con

```
#exit
```

#### Información del sistema operativo

```
>docker logs 08c86d20b5bc
```

## **Instalar Git en Fedora36**

1. Vamos a actualizar nuestro GNU/Linux Fedora 36

```
>sudo dnf upgrade --refresh
```

2. Luego ejecutamos el siguiente comando para instalar git

```
>sudo dnf install git -y
```

3. Colocando nombre de usuario.

```
>git config --global user.name "mi nombre"4-
```

Colocando el correo electrónico del usuario.

```
>git config --global user.email "miusuario@gmail.com"
```

Creando directorio para GIT

```
>mkdir Git-directory -p
```

Vamos a home/w2k31984 y seleccionamos la carpeta creada luego damos click derecho abrir terminal

Luego inicializamos git

```
>git init
```

```
>git config --list
```

## **Practica.**

1-Clonaremos el libro de Mastering Lightning Network en nuestra carpeta de Git-directory

```
>mkdir lnbook
```

```
>cd lnbook
```

```
>git clone https://github.com/lnbook/lnbook.git
```

2-Hacemos pull en docker lnbook

```
>docker pull lnbook/bitcoind
```

Para la practica crearemos localmente el siguiente directorio local

```
>mkdir code
```

```
>cd code
```

```
>mkdir docker
```

```
>cd docker
```

Vamos al directorio local creado y ejecutamos el siguiente comando de docker

```
> docker run -it --name bitcoind lnbook/bitcoind
```

Explicación : bitcoind se pone en marcha y mina 101 bloques simulados para iniciar la cadena. Esto es porque bajo las reglas de consenso de Bitcoin, el bitcoin recién minado no es gastable hasta que hayan transcurrido 100 bloques. Al minar 101 bloques, hacemos que la base de monedas del primer bloque sea gastable. Después de esa actividad minerainicial, se minan 6 nuevos bloques cada 10 segundos para que la cadena siga avanzando.

## **Interacción con el contenedor**

Ahora corremos

```
> docker exec -it bitcoind /bin/bash
```

```
>ps x
```

```
#exit
```

Por comodidad, el comando bitcoin-cli tiene un alias "cli" que pasa la configuración correcta. Así que vamos a ejecutarlo para preguntar a Bitcoin Code sobre el blockchain.

Ejecutamos cli getblockchaininfo:

```
>docker exec bitcoind cli getblockchaininfo
```

El comando cli en el contenedor bitcoind nos permite emitir comandos RPC al nodo Bitcoin Core y obtener resultados codificados en JavaScript Object Notation (JSON).

```
>docker exec bitcoind bash -c "cli getblockchaininfo | jq .blocks"
```

## Construcción de c-lightning como contenedor Docker

La distribución del software c-lightning tiene un contenedor Docker, pero está diseñado para ejecutar c-lightning en sistemas de producción y junto a un nodo bitcoind. Utilizaremos un contenedor algo mas simple configurado para ejecutar c-lightning con fines de demostración.

Vamos a sacar el contenedor c-lightning del repositorio Docker Hub del libro:

```
>docker pull lnbook/c-lightning
```

Ahora iremos al directorio de lnbook en code/docker y ejecutaremos la siguiente instrucción de docker

```
>docker build -t lnbook/c-lightning c-lightning
```

Creando una red con docker de LN

```
>docker network create lnbook
```

```
>docker network ls
```

## Ejecución de los contenedores bitcoind y c-lightning

1. Vamos a ejecutar la siguiente instrucción

```
>docker run -it --network lnbook --name bitcoind lnbook/bitcoind
```

2. Deberías ver que bitcoind se inicia y comienza a minar bloques cada 10 segundos. Déjalo funcionando y abre una nueva ventana de terminal para iniciar c-lightning.

Utilizamos un comando docker run similar con los argumentos de red y nombre para iniciar c-lightning de la siguiente manera:

```
>docker run -it --network lnbook --name c-lightning lnbook/c-lightning
```

3. Como demostramos con el contenedor bitcoind, podemos emitir comandos a nuestro contenedor c-lightning en otra terminal para extraer información, abrir canales, etc. El comando que nos permite emitir instrucciones de línea de comandos al nodo c-lightning se llama lightning-cli. Este comando lightning-cli también tiene el alias de cli dentro de este contenedor. Para obtener la información del nodo c-lightning, utilice el siguiente comando docker exec en otra ventana de terminal:

```
>docker exec c-lightning cli getinfo
```

Ahora tenemos nuestro primer nodo Lightning funcionando en una red virtual y comunicándose con una blockchain de Bitcoin de prueba.

## **Copiar el código fuente de c-lightning**

Actualizando paquetes de fedora36

```
>sudo dnf upgrade --refresh  
>sudo dnf update
```

A continuación, copiaremos la última versión de c-lightning desde el repositorio de código fuente. Para ello, utilizaremos el comando git clone, que clona una copia controlada de la versión en nuestra máquina local, lo que le permite mantenerla sincronizada con los cambios posteriores sin tener que descargar todo el repositorio de nuevo lo haremos ejecutando desde nueva terminal la siguiente instrucción:

```
>git clone --recurse https://github.com/ElementsProject/  
lightning.gitAhora iremos al directorio creado  
>cd lightning
```

No necesitamos cambiar ninguno de los valores por defecto para este ejemplo. Por lo tanto, ejecutamos configure de nuevo sin ninguna opción para utilizar los valores predeterminados:

```
>./configure
```

A continuación, utilizamos el comando make para construir las bibliotecas, componentes y ejecutables del proyecto c-lightning. Esta parte tardará varios minutos en completarse y utilizará mucho la CPU y el disco de tu ordenador.

```
>make
```

El contenedor Docker LND

Podemos sacar el contenedor Docker de ejemplo de LND del repositorio Docker Hub del libro:Ejecutamos la instrucción en una terminal nueva

```
>docker pull lnbook/lnd
```

Luego vamos a la carpeta del libro a code/docker, y ejecutamos la siguiente instrucción

```
>docker build -t lnbook/lnd lnd
```

### **Ejecución de los contenedores bitcoind y LND**

1-Ejecutamos la instrucción siguiente en una terminal nueva cambiando el nombre por bitcoindln:

```
>docker run -it --network lnbook --name bitcoindln lnbook/bitcoind
```

2-A continuación, iniciamos el contenedor LND que acabamos de construir. Como se ha hecho antes,tenemos que adjuntarlo a la red lnbook y darle un nombre en una terminal nueva:

```
>docker run -it --network lnbook --name lnd lnbook/lnd
```

3. Vemos la información del nodo utilizando el comando docker exec en otra nueva terminal:

```
>docker exec lnd cli getinfo
```

4. Si lo desea, puede ejecutar cualquier combinación de nodos LND y c-lightning en la misma red Lightning. Por ejemplo, para ejecutar un segundo nodo de LND, deberá emitir el comando docker runcon un nombre de contenedor diferente, en una nueva terminal:

```
>docker run -it --network lnbook --name lnd2 lnbook/lnd
```

## **Instalación de LND desde el código fuente**

1. Instalamos golang con la siguiente instrucción

```
>sudo dnf install -y go
```

2. Ver la version instalada de go

```
>go version
```

3. Colocando go en la variable de entorno

```
>export GOPATH=~/gocode
```

```
>export PATH=$PATH:$GOPATH/bin
```

## **Copiar el código fuente de LND**

1. En una terminal nueva colocamos la siguiente instrucción:

```
> git clone https://github.com/lightningnetwork/lnd
```

2. Vamos a la carpeta creada y abrimos una terminal nueva desde ahi

```
>make install
```

3. Ejecutamos la instrucción siguiente dentro de carpeta LN

```
>make && make install
```

## **Eclair Lightning Node Project**

1-Vamos a sacar el contenedor Eclair del libro del repositorio Docker Hub:

```
>docker pull lnbook/eclair
```

## **Ejecución de los contenedores bitcoind y Eclair**

1. Para iniciar bitcoind en la red lnbook, utilizamos docker run así:

```
> docker run -it --network lnbook --name bitcoind lnbook/bitcoind
```

2. Iniciamos el contenedor Eclair que acabamos de construir. Tendremos que adjuntarlo a la red lnbooky darle un nombre, al igual que hicimos con los otros contenedores en nueva terminal:

```
>docker run -it --network lnbook --name eclair lnbook/eclair
```

3. Usando el comando docker exec en otra ventana de terminal, obtenemos la información del nododesde Eclair:

```
>docker exec eclair cli getinfo
```

4. Creamos otro contenedor con la wallet eclair

```
>docker run -it --network lnbook --name eclair2 lnbook/eclair
```



## Uso de docker-compose para orquestar contenedores Docker

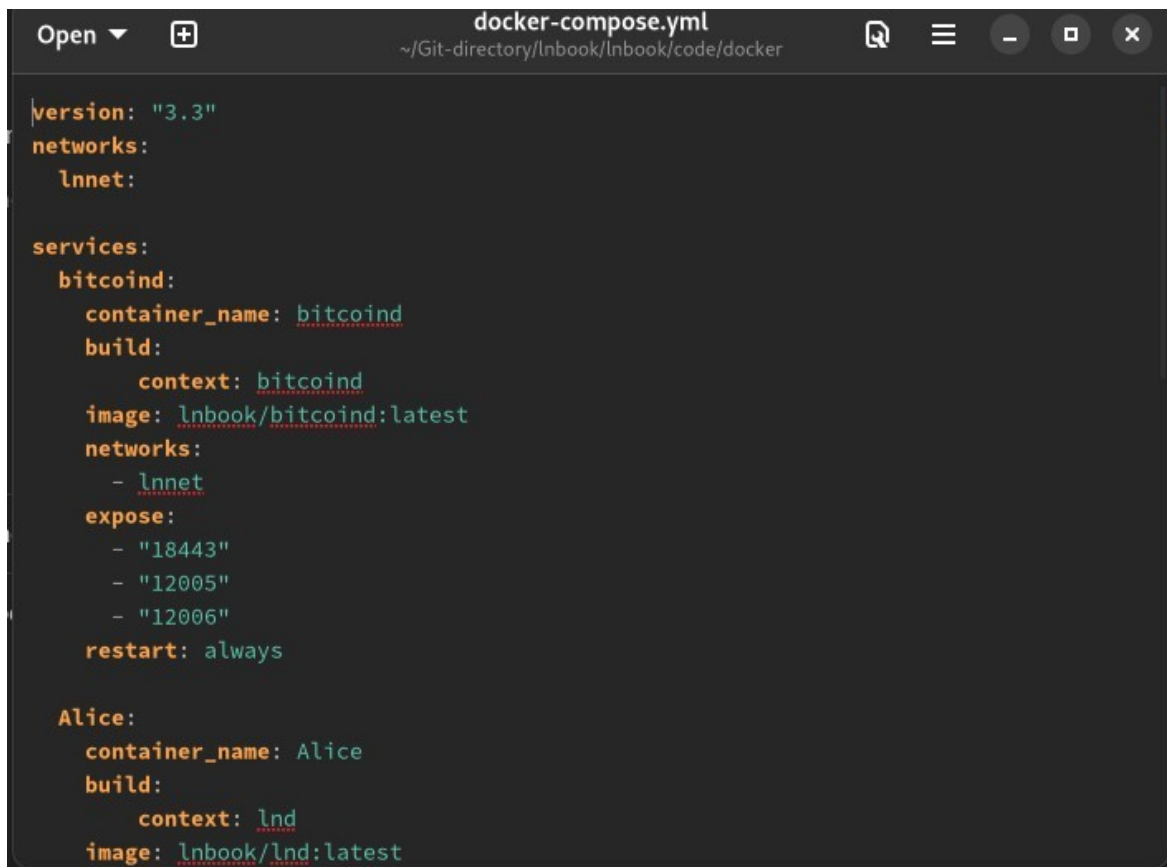
1. Este comando nos permite especificar una aplicación compuesta por varios contenedores y ejecutar la aplicación lanzando todos los contenedores que cooperan juntos en una terminal nueva ejecutaremos la instrucción siguiente.

>docker-compose version

### Configuración de docker-compose

El archivo de configuración para docker-compose se encuentra en el directorio code/docker y se llama docker-compose.yml. Contiene una especificación para una red y cada uno de los cuatro contenedores. La parte superior tiene el siguiente aspecto:

\*Validaremos abriendo el archivo en la ruta donde nos especifica esta la carpeta code/docker



```
version: "3.3"
networks:
  lnnet:

services:
  bitcoind:
    container_name: bitcoind
    build:
      context: bitcoind
    image: lnbook/bitcoind:latest
    networks:
      - lnnet
    expose:
      - "18443"
      - "12005"
      - "12006"
    restart: always

  Alice:
    container_name: Alice
    build:
      context: lnd
    image: lnbook/lnlnd:latest
```

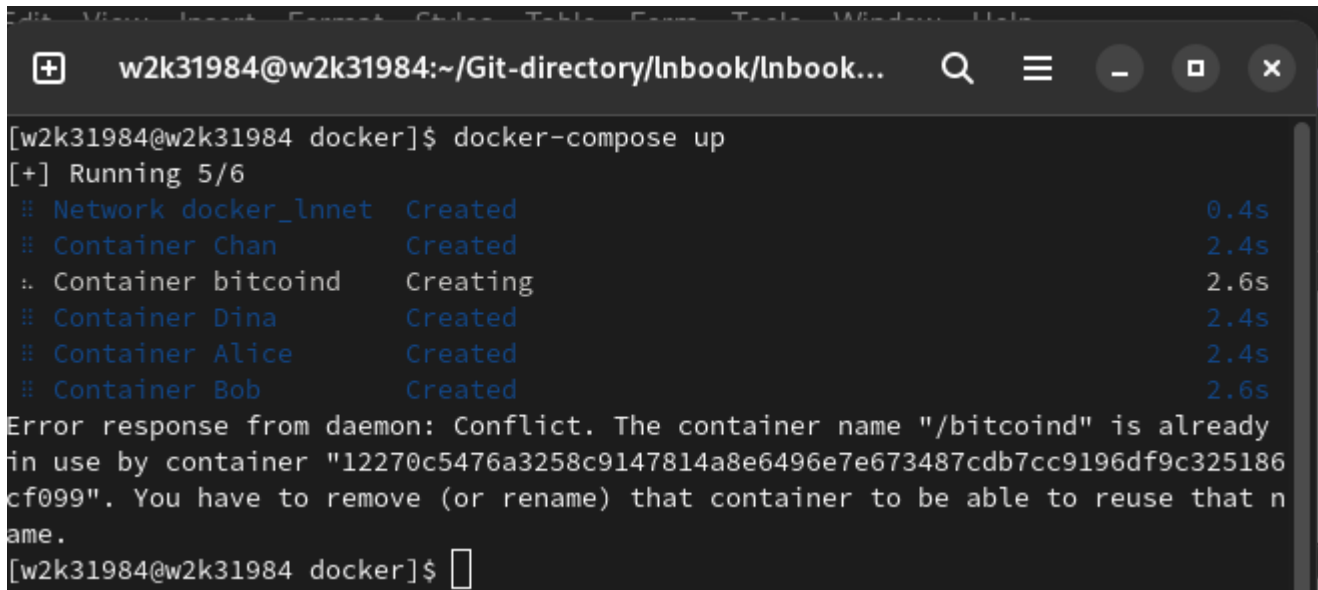
2. Para iniciar el ejemplo, cambiamos al directorio que contiene el archivo de configuración `docker-compose.yml` y emitimos el comando `docker-compose up`:

\*Vamos donde esta nuestra carpeta `code/docker`

Abrimos una terminal al estar dentro y ejecutamos la siguiente instrucción

>`docker-compose up`

\*Si envía el siguiente mensaje sera entonces de crear con otro nombre el contenedor `bitcoind` Y correr nuevamente la instrucción de `docker`.



```
w2k31984@w2k31984:~/Git-directory/lnbook/lnbook...
[w2k31984@w2k31984 docker]$ docker-compose up
[+] Running 5/6
  :: Network docker_lnnet    Created                                0.4s
  :: Container Chan          Created                                2.4s
  :: Container bitcoind       Creating                             2.6s
  :: Container Dina          Created                                2.4s
  :: Container Alice         Created                                2.4s
  :: Container Bob           Created                                2.6s
Error response from daemon: Conflict. The container name "/bitcoind" is already
in use by container "12270c5476a3258c9147814a8e6496e7e673487cdb7cc9196df9c325186
cf099". You have to remove (or rename) that container to be able to reuse that n
ame.
[w2k31984@w2k31984 docker]$
```

Con esta instrucción creamos el contenedor nuevo cambiando el nombre a `bitcoind1` debemos ejecutar esta instrucción en una terminal nueva.

>`docker run -it --network lnbook --name bitcoind1 lnbook/bitcoind`

Corremos la instrucción de `docker composer` nuevamente

>`docker-compose up`

\*Si tenemos algún otro contenedor corriendo debemos eliminarlo de `docker` y ejecutar la instrucción nuevamente.

Una vez este corriendo los contenedores y los nodos podemos ver los registros de un solo contenedor, puedes hacerlo en otra ventana de terminal utilizando el comando `docker-compose logs` con la bandera `f` (follow) y el nombre del contenedor específico en una terminal dentro de la carpeta `code/docker`:

>`docker-compose logs -f Alice`

```

=> => writing image sha256:a3bf5c417072f327200c011d53ecdb7d731b17ba093de 0.1s
=> => naming to docker.io/lnbook/lnd 0.1s
[w2k31984@w2k31984 docker]$ docker-compose logs -f Alice
Alice | Waiting for bitcoind to start...
Alice | .....Waiting for bitcoind to mine blocks...
Alice | Starting lnd...
Alice | Startup complete
Alice | Funding lnd wallet
Alice | % Total % Received % Xferd Average Speed Time Time Time
Current
Alice | Dload Upload Total Spent Left
Speed
100 295 100 116 100 179 30 46 0:00:03 0:00:03 --:--:-- 76
Alice | {"result":"662cd701c1e39d38ef6cb9c9f98bd772190f36a78483fc6c471767086f02
cee6","error":null,"id":"lnd-run-container"}

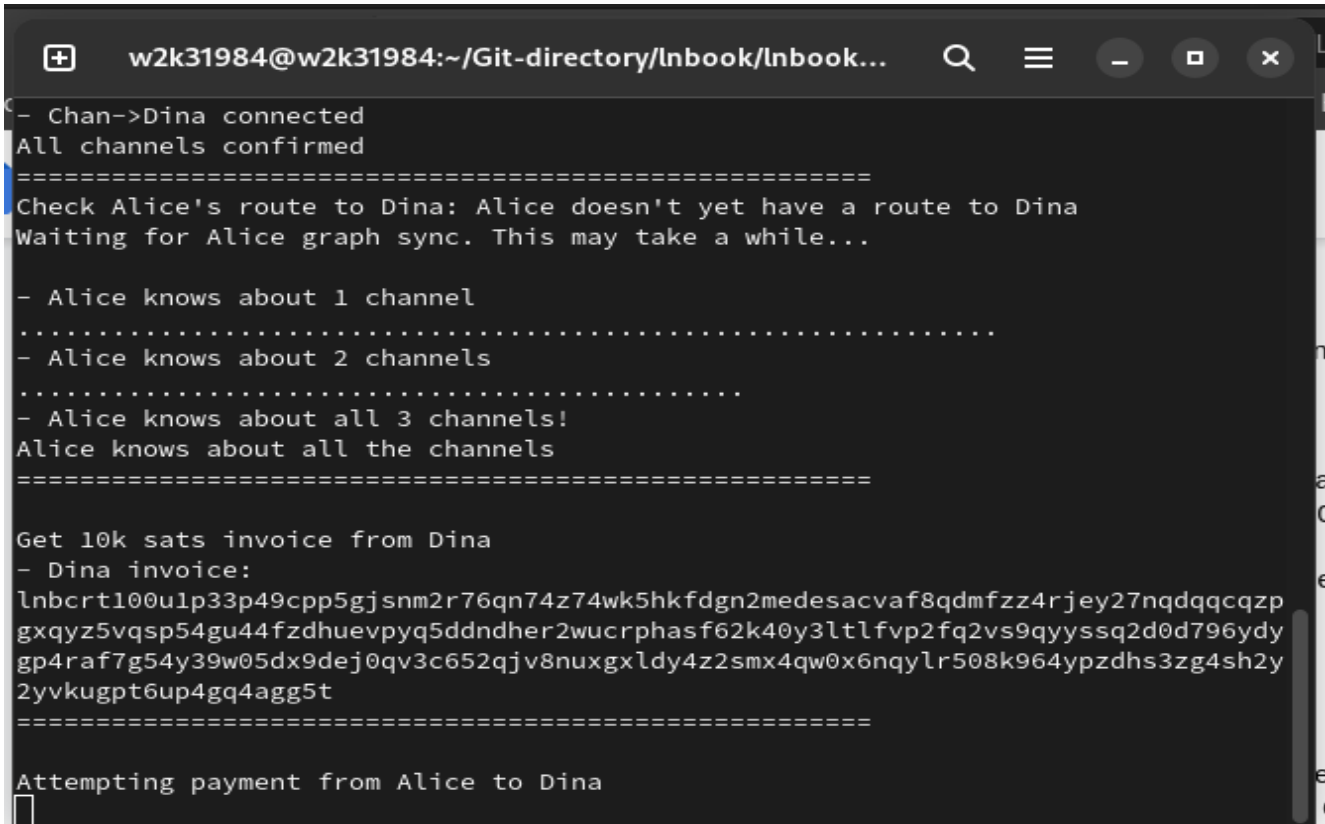
```

Vamos a ejecutar el script para ver su efecto, y luego veremos cómo funciona internamente. Usamos bash para ejecutarlo como un comando esto dentro de la carpeta code/docker:

>bash run-payment-demo.sh

## Practica

1. Actualizar los balances del diagrama después de que Alice hace el pago.
- 2.



```
w2k31984@w2k31984:~/Git-directory/lnbook/lnbook...
- Chan->Dina connected
All channels confirmed
=====
Check Alice's route to Dina: Alice doesn't yet have a route to Dina
Waiting for Alice graph sync. This may take a while...

- Alice knows about 1 channel
.....
- Alice knows about 2 channels
.....
- Alice knows about all 3 channels!
Alice knows about all the channels
=====

Get 10k sats invoice from Dina
- Dina invoice:
lnbcrt100ulp33p49cpp5gjsnm2r76qn74z74wk5hkfdgn2medesacvaf8qdmfzz4rjey27nqdqqcqzp
gxqyz5vqsp54gu44fzdhuevpyq5ddndher2wucrphasf62k40y3ltlfvp2fq2vs9qyysq2d0d796ydy
gp4raf7g54y39w05dx9dej0qv3c652qjv8nuxgxlidy4z2smx4qw0x6nqylr508k964ypzdhs3zg4sh2y
2yvkugpt6up4gq4agg5t
=====

Attempting payment from Alice to Dina
█
```

2. Dina debe generar un nuevo invoice de 20K que debe ser pagado por Bob. Para probar que hicisteel pago toma un pantallazo de la preimagen
3. Abrir un canal de Dina a Alice de 500.000 satoshis. Para probar que abriste el canal toma un pantallazo del comando *listchannels* desde el nodo de Dina

### Documentación:

-<https://drive.google.com/drive/folders/1kTBj8fAfxVRSpWq3AxK3OIBUvm2zHkvD?usp=sharing>

-<https://www.itzgeek.com/how-tos/linux/fedora-how-tos/install-docker-on-fedora.html>

-<https://techviewleo.com/install-use-docker-desktop-on-fedora/>

-<https://www.linuxcapable.com/how-to-install-git-on-fedora-36-linux/>

-<https://github.com/lightningnetwork/lnd/blob/master/docs/INSTALL.md>

**Documento creado por Cristian Parada Alumno de Torogoz Dev y La Libreria de Satoshi**