Universidad Nacional de Quilmes

Laboratorio de Redes y Sistemas Operativos

Kubo (Una implementación de IPFS en Go)

Integrantes:

- Maximiliano Borrajo (maximilianoborrajo0@gmail.com)
- Franco Orizonte (francoorizonte@gmail.com)

Docente:

- José Luis Di Biase

1. Descripción del proyecto

Instalar, configurar y poner en funcionamiento un nodo IPFS utilizando Kubo (implementación oficial en Go, instalada como binario nativo, sin Docker), demostrando de forma funcional:

1. Instalación y configuración del nodo.
2. Manejo de archivos, bloques y pinning.
3. Uso de la WebUI.
4. Publicación de un sitio web estático accesible desde gateways públicos.
5. Direccionamiento mutable con IPNS.
6. Resolución mediante DNSLink sobre un dominio real.

¿Qué es IPFS? ¿Qué es kubo?

IPFS (InterPlanetary File System) es un protocolo P2P de almacenamiento y distribución de archivos donde el contenido se direcciona por su hash (CID) en lugar de por ubicación. El contenido es inmutable (cambiar un byte cambia el CID), verificable (el hash prueba que no fue alterado) y no depende de un servidor central — cualquier nodo que tenga el contenido puede servirlo. Para encontrar quién tiene qué, usa una DHT distribuida entre los nodos.

Kubo es la implementación de referencia del protocolo, escrita en Go. El cual nos va a ofrecer la funcionalidades que necesitamos para la puesta a punto.

DHT (Distributed Hash Table) es una tabla hash distribuida entre todos los nodos de la red: un key-value store donde ningún nodo tiene la tabla completa, cada uno guarda una porción y sabe a quién preguntar por el resto.

2. Entorno de trabajo

Requerimientos mínimos:
Sistema Operativo: Windows, MacOS, Linux, FreeBSD o OpenBSD.
RAM: 6GB
CPU: 2 CPU cores
Almacenamiento: IPFS usa por defecto 12MB de espacio en disco.

Recursos utilizados:
En máquina virtual (Oracle VirtualBox)
Sistema Operativo: Debian GNU/Linux (trixie)
RAM: 2GB
Procesador: AMD Ryzen 7 7730U with Radeon Graphics (2.00 GHz) (1 CPU core)
Almacenamiento: 20 GB
Red: adaptador puente (bridged), para que la VM obtenga IP propia en la red local y el nodo IPFS pueda recibir conexiones entrantes sin la capa adicional de NAT de VirtualBox.

3. Instructivo Paso a Paso de lo Ocurrido (Ejecución)

3.1 Instalación de Kubo

Para instalar Kubo, la documentación nos ofrece un paso a paso con cada sistema operativo, para este caso utilizaremos Linux.

1. Descargar el binario de Linux desde dist.ipfs.tech

```
wget https://dist.ipfs.tech/kubo/v0.42.0/kubo_v0.42.0_linux-amd64.tar.gz
```

2. Descomprimir el archivo

```
tar -xvzf kubo_v0.42.0_linux-amd64.tar.gz

> x kubo/install.sh
> x kubo/ipfs
> x kubo/LICENSE
> x kubo/LICENSE-APACHE
> x kubo/LICENSE-MIT
> x kubo/README.md
```

3. Entrar a la carpeta kubo

```
cd kubo
```

4. Ejecutar el script de instalación

```
sudo bash install.sh

> Moved ./ipfs to /usr/local/bin
```

5. Verificar que Kubo se instaló correctamente

```
ipfs --version

> ipfs version 0.42.0
```

3.2 Inicialización del nodo

1. Abrir una terminal
2. Inicializar repositorio

```
ipfs init
```

La respuesta será similar a esto

```
initializing ipfs node at /Users/<user>/.ipfs
generating 2048-bit RSA keypair...done
peer identity: Qm...
```

```
to get started, enter:

  ipfs cat /ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWEz79ojWnPbdG/readme
```

Esto básicamente genera unos archivos de configuración base para empezar a levantar un nodo.

En caso de estar con internet público, se puede inicializar con perfil de servidor, lo cual deshabilita el descubrimiento por red local y rechaza conexiones a tu nodo desde conexiones no localizables.

```
ipfs init --profile server
```

3.3 Conexión a la red nodos

Ahora si, para poner el nodo en línea, ejecutar:

```
ipfs daemon

> Initializing daemon...
> API server listening on /ip4/127.0.0.1/tcp/5001
> Gateway server listening on /ip4/127.0.0.1/tcp/8080
```

Esto mantendrá el nodo en línea hasta que lo pares, si lo haces desconectara el nodo de la red ipfs, por ahora no lo hagas.

Para terminar de verificar si estas conectado a la red, abre otra terminal si eliminar la anterior, y ejecuta el siguiente comando para ver las direcciones IPFS de tus pares:

```
ipfs swarm peers

>/ip4/104.131.131.82/tcp/4001/p2p/QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ
>/ip4/104.236.151.122/tcp/4001/p2p/QmSoLju6m7xTh3DuokvT3886QRYqxAzb1kShaanJgW36yx
>/ip4/134.121.64.93/tcp/1035/p2p/QmWHyrPWQnsz1wxHR219ooJDYTvxJPyZuDUPSDpdsAovN5
>/ip4/178.62.8.190/tcp/4002/p2p/QmdXzZ25cyzSF99csCQmmPZ1NTbWTe8qtKFaZKpZQPdTFB
```

Las direcciones mostradas se componen de una <transport address> (i.e. /ip4/104.131.131.82/tcp/4001) y un <hash-of-public-key> (i.e. QmSoLju6m7xTh3DuokvT3886QRYqxAzb1kShaanJgW36yx), lo que da como resultado una dirección de la forma <transport address>/p2p/<hash-of-public-key>.

3.3 Manejo de archivos y bloques

1. Agregar archivos

Primero vamos a navegar al directorio que tenga el archivo o carpeta que queramos compartir, en este caso nos movemos a `~/Documents`

```
cd ~/Documents
```

Luego listamos los contenidos del directorio para asegurarnos que estamos en el lugar correcto

```
ls
```

Si ya tenemos un archivo elegido, ejemplo `hello.txt`, ya podríamos subir el archivo en la red ipfs, pero en nuestro caso vamos a crear un archivo de cero y subirlo.

Ejecutamos este comando para crear nuestro archivo

```
echo "meow" > meow.txt
```

Luego simplemente ejecutamos este comando para subir nuestro archivo

ipfs add <nombre del archivo>

```
ipfs add meow.txt
> added QmabZ1pL9npKXJg8JGdMwQMJo2NCVy9yDVYjhiHK4LTJQH meow.txt
6 B / 6 B [============================================================]
100.00
```

Su CID del archivo será distinto al nuestro si eligió un archivo distinto.

Ya hemos añadido un archivo a IPFS y ya puede ser compartido con otros usuarios de la red

2. Recuperar un archivo

Primero vamos a navegar al directorio donde queramos guardar el archivo, en este caso nos movemos a `~/Documents`

```
cd ~/Documents
```

Para obtener contenido a través de IPFS, debemos indicarle al ipfs daemon qué CID queremos. En este caso, queremos bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a.

```
ipfs get bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
Saving file(s) to
bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
1.76 KiB / 1.76 KiB [============================================]
100.00% 0s
```

Puede que tu nodo IPFS tarde unos instantes en encontrar los datos que buscamos.

Una vez hecho esto, ya hemos recuperado un archivo de la red ipfs y guardamos una copia en nuestra computadora. Por lo que ahora también somos hosts de este archivo, y otros pueden obtenerlo de nosotros.

3. Ver un archivo

Para ver el archivo anteriormente descargado ejecutamos el siguiente comando

```
ipfs cat bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
```

Te va a lanzar un error asi:

```
Error: this dag node is a directory
```

Esto es porque este CID apunta a un directorio, no un el archivo que descargamos, así que para ver el contenido de este directorio hacemos

```
ipfs refs bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
```

Nos dará algo asi:

```
bafkreig24ijzqxj3cxdp6yh6ia2ysxzvxsfnd6rzahxxjv6ofcuix52wtq
```

El comando `ref` devuelve el CID que apunta al archivo dentro del directorio. Ahora podemos usar `ipfs cat` con este nuevo CID para verlo:

```
ipfs cat bafkreig24ijzqxj3cxdp6yh6ia2ysxzvxsfnd6rzahxxjv6ofcuix52wtq
```

Lo que va a devolver algo asi:

```
MMMMMMMMMN0xo;';ox0NMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMWXOdoloxkkkxolodOXWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMN0xdoodkOOOOOOOOOkdoodx0NMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MKo;;oxOOOOOOOOOOOOOOOOxo;;oKMMMMMXOKMMMN0kkkkkO0NWMMXOkkkkkkOXMMWKkdddddk0NMMMM
Wd...':okOOOOOOOOOOOOOko:'...dWMMMWd.lWMM0,.;ccc:;,oXWd.'clllco0WO:,:loolcckWMMM
Wdckdc,..;lxOOOOOOOxl;..';lo;dWMMMWo.lWMM0''0MMMWK:.oNo.lWMMMMMMX:.dWMMMMMWWMMMM
WdcOKK0ko;'.,:c:c:,..,:oxxxd;dWMMMWo.lWMM0''0MMMMNc.oNo.lNWWWWWMNl.;x0XNWMMMMMMM
WdcOKKKKKKOd:.    .,cdxxxxxxd;dWMMMWo.lWMM0'.coool;'cKWo..clllldXMNkl:;;;:lxXMMMM
WdcOKKKKKKKK0l. .:dxxxxxxxxd;dWMMMWo.lWMM0'.cooodkKWMWo.:0K000KWMMMMWNK0xc.,0MMM
WdcOKKKKKKKK0: ,dxxxxxxxxxd:dWMMMWo.lWMM0''0MMMMMMMMWo.lWMMMMMMMMMMMMMMX:.dWMM
Wd;xKKKKKKKK0: ,dxxxxxxxxxl,dWMMMWo.lWMM0''0MMMMMMMWo.lWMMMMMWOdk0KXXKd.,0MMM
Mk',d0KKKKKK0: ,dxxxxxxxdc.'kMMMMMO:xWMMKllXMMMMMMMMWk:kWMMMMMWOlccccccccdKWMMM
MWKkdooxOKKKK0: ,dxxxxdlclokKWMMMMWWWMMMMWWMMMMMMMMMMMWMMMMMMMMMMMWWNNNNWMMMMMM
MMMMWNOxdodk00: ,ddoccldONWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMWKkdol' .:lokKWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMWKd'.'dKWMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
```

Es importante tener en cuenta que `ipfs cat` solo funciona con archivos de texto plano. Como se demostró anteriormente, intentar mostrar el contenido de un directorio generará un error. Si intenta mostrar el contenido de un archivo de imagen o de texto que no sea de texto plano, su terminal se llenará de líneas ilegibles.

4. Fijar un archivo

   Para fijar un archivo, de modo de no perderlo por el garbage collector usamos

   ```
   ipfs pin add bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
   ```

   Devolverá algo como:

   ```
   pinned bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
   recursively
   ```

   Por defecto, los objetos que se recuperan a través de IPFS no se fijan a tu nodo. Si deseas evitar que los archivos se eliminen mediante el garbage collector, debes fijarlos. La fijación del ejemplo fue recursiva, lo que indica que es un directorio.

5. Eliminar una fijación

   Para dejar de mantener en nuestro nodo un archivo, podemos removerlo.

   Primero listamos nuestro archivos fijados, ya que si no lo estuvieran, no hace falta eliminarlos manualmente, de eso se encarga en garbage collector.

   ```
   ipfs pin ls
   ```

   Nos devolverá el listado de CID fijados, algo así

   ```
   QmPZ9gcCEpqKTo6aq61g2nXGUhM4iCL3ewB6LDXZCtioEB indirect
   QmQGiYLVAdSHJQKYFRTJZMG4BXBHqKperaZtyKGmCRLmsF indirect
   QmU5k7ter3RdjZXu3sHghsga1UQtrztnQxmTL22nPnsu3g indirect
   QmUNLLsPACCz1vLxQVkXqqLX5R1X345qqfHbsf67hvA3Nn recursive
   QmYCvbfNbCwFR45HiNP45rwJgvatpiW38D961L5qAhUM5Y indirect
   QmejvEPop4D7YUadeGqYWmZxHhLc4JBUCzJJHWMzdcMe2y indirect
   QmQ5vhrL7uv6tuoN9KeVBwd4PwfQkXdVVmDLUZuTNxqgvm indirect
   QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc recursive
   QmQy6xmJhrcC5QLboAcGFcAE1tC8CrwDVkrHdEYJkLscrQ indirect
   ```

   Si sabemos el CID, con ese nos alcanza para eliminarlo, pero si no, podemos volver a añadir el archivo que queramos eliminar, lo cual no lo vuelve añadir, sino que solamente nos devuelve el CID de dicho archivo.

   Vamos a hacer esto mismo con el archivo meow.txt previamente. Este archivo aparece como fijado en nuestro nodo local porque nosotros mismos lo subimos. Los archivos que subamos nosotros se fijan por defecto en nuestro nodo local.

```
cd ~/Documents
ipfs add meow.txt
> added QmabZ1pL9npKXJg8JGdMwQMJo2NCVy9yDVYjhiHK4LTJQH meow.txt
6 B / 6 B [============================================================]
100.00
```

Ahora con el CID, eliminamos el archivo fijado.

```
ipfs pin rm QmabZ1pL9npKXJg8JGdMwQMJo2NCVy9yDVYjhiHK4LTJQH
```

Nos devolverá algo como:

```
unpinned QmabZ1pL9npKXJg8JGdMwQMJo2NCVy9yDVYjhiHK4LTJQH
```

Pero eso solo elimina la fijación, para eliminarlo de nuestro modo, ejecutamos el garbage collector manualmente

```
ipfs repo gc
```

Nos devolverá algo como:

```
removed bafybeif2ewg3nqa33mjokpxii36jj2ywfqjpy3urdh7v6vqyfjoocvgy3a
removed bafkreieceevgg2auxo4u3rjgeiqfr4ccxh6ylkgxt2ss6k2leuad5xckxe
removed bafkreiblcvcr7letdbp2k2thkbjyunznrwq3y6pyoylzaq4epawqcca2my
[...]
```

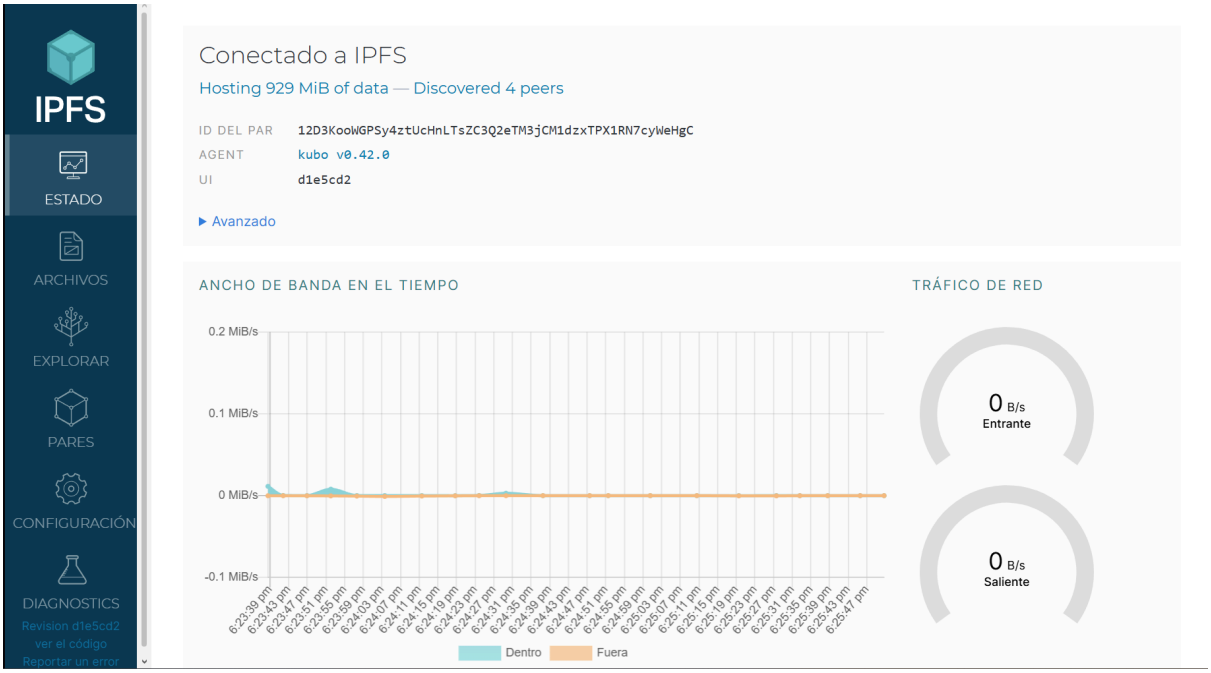
Si desea eliminar el contenido del almacenamiento local de su equipo, deberá encontrar dónde se guardó y eliminarlo mediante el método de eliminación habitual.

3.4 WebUI

Para Ingresar a la webUi, si vamos a la terminal que dejamos levantada al ejecutar ipfs daemon, veremos

```
WebUI: http://127.0.0.1:5001/webui
```

Ingresando a la misma podrán ver la información de su nodo



La consola web muestra los archivos que se encuentran en su Sistema de Archivos Mutable (MFS). MFS es una herramienta integrada en la consola web que le permite navegar por los archivos IPFS de la misma manera que lo haría en un sistema de archivos estándar basado en nombres.

1. Ingrese http://127.0.0.1:5001/webui en su navegador para ver la consola web.

En el menú de la barra lateral izquierda, haga clic en Archivos. Se mostrará un directorio vacío, junto con el siguiente mensaje: ¡Aún no hay archivos aquí!

2. Vuelve a la ventana de tu terminal original.
3. Utilizando el CID <CID> obtenido al agregar meow.txt a su nodo en el paso anterior, copie los archivos al MFS.

```
ipfs files cp /ipfs/<CID> /meow.txt
```

4. En tu navegador, actualiza la página de Archivos. La lista de archivos mostrará meow.txt.

3.5 Publicación de un sitio web estático

Ahora vamos a mostrar con todo lo mostrado cómo publicar un sitio web en ipfs.

1. Abrimos una terminal y ejecutamos

```
mkdir mi-sitio
cd mi-sitio
```

2. Creamos el archivo index.html

```
cat > index.html << 'EOF'
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Mi sitio en IPFS</title>
    <style>
      body {
        margin: 15px auto;
        max-width: 650px;
        font-family: sans-serif;
        font-size: 1.5em;
        color: #fff;
        background: #444;
      }
    </style>
  </head>
  <body>
    <h1>Hola desde IPFS</h1>
    <p>Este sitio se sirve desde un nodo Kubo, sin servidor central.</p>
  </body>
</html>
EOF
```

3. Subimos el directorio a la red

```
cd ..
ipfs add -r --cid-version 1 mi-sitio

> added bafkreialdo3ba6df3xr4jhys3maabgtdgcv6xdlq6hffgb35qhwlkak5da
mi-sitio/index.html
> added bafybeihnoabidyob44mrbcbrembmyaqbybuztl2rgtxbz4y5o5da3ta3le mi-sitio
```

Se agrega el directorio completo de forma recursiva con -r. El flag --cid-version 1 genera CIDs v1 (base32, minúsculas), necesarios para que funcione el acceso por subdominio en gateways públicos.

El CID que importa es el último (el del directorio): es el hash del nodo raíz del DAG que contiene todo el sitio.

4. Ahora ya podemos acceder al sitio web desde un navegador.

```
http://127.0.0.1:8080/ipfs/<CID>/
```

También podemos acceder desde gateways públicos que permite ipfs

```
https://ipfs.io/ipfs/<CID>/
```

La primera carga puede tardar unos minutos: el gateway todavía tiene que descubrir tu nodo en la DHT.

5. Para mantener el sitio disponible hay varias opciones
    - Nodo propio siempre encendido: alcanza con dejar el daemon corriendo.
    - Segundo nodo propio: en la otra máquina, ipfs pin add <CID> para replicar el contenido.
    - Servicio de pinning remoto (Pinata, Filebase): un tercero mantiene el contenido pinneado.

```
ipfs pin remote service add <nickname> <endpoint> <accessToken>
ipfs pin remote add --service=<nickname> --name=mi-sitio <CID>
```

3.6 IPNS: identificador mutable

Como el CID es inmutable, cualquier cambio a nuestra página web generaría un CID nuevo, por lo tanto vamos a registrarlo en IPNS lo cual generará un CID único para nuestros archivos.

1. Usando el CID de la web que subimos previamente, lo publicamos

```
ipfs name publish /ipfs/bafybeihnoabidyob44mrbcbrembmyaqbybuztl2rgtxbz4y5o5da3ta3le

> Published to k51qzi5uqu5dgy6fu9073kabgj2nuq3qyo4f2rcnn4380z6n8i4v2lvo8dln6l:
/ipfs/bafybeihnoabidyob44mrbcbrembmyaqbybuztl2rgtxbz4y5o5da3ta3le
```

    k51... es la clave pública o el nombre IPNS del IPFS que está utilizando. Ahora puede modificar el archivo repetidamente y, aunque el CID cambie al modificarlo, podrá seguir accediendo a él con esta clave.

2. Puedes ver tu web accediendo a https://ipfs.io/ipns/<CID>

3. Modifica tu archivo, agrégalo a IPFS y actualiza tu IPNS:

```
cd mi-sitio
cat > index.html << 'EOF'
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8" />
    <title>Mi sitio en IPFS</title>
    <style>
      body {
        margin: 15px auto;
        max-width: 650px;
        font-family: sans-serif;
        font-size: 1.5em;
        color: #fff;
        background: #444;
      }
    </style>
  </head>
  <body>
```

```
    <h1>Hola de nuevo desde IPFS</h1>
    <p>Este sitio se sirve desde un nodo Kubo, sin servidor central.</p>
  </body>
</html>
EOF

cd ..
ipfs add -r --cid-version 1 mi-sitio

> added bafkreidbbor7mvra2xzzl4kmr2sxrtkzaxlzs6rsr5ktgmbtousuzrhlxq mi-sitio
> 17 B / 17 B [=========================================================] 100.00%

ipfs name publish /ipfs/bafkreidbbor7mvra2xzzl4kmr2sxrtkzaxlzs6rsr5ktgmbtousuzrhlxq

> Published to k51qzi5uqu5dgy6fu9073kabgj2nuq3qyo4f2rcnn4380z6n8i4v2lvo8dln6l:
/ipfs/bafkreidbbor7mvra2xzzl4kmr2sxrtkzaxlzs6rsr5ktgmbtousuzrhlxq
```

4. La misma URL de siempre muestra la versión nueva. Los gateways públicos pueden tardar en reflejar el cambio; el gateway local y ipfs cat /ipns/<nombre>/index.html lo muestran al instante. En cualquier caso, primero probar con:

```
curl -L
https://ipfs.io/ipns/bafybeiaotxdyji4bwktyq53jyfb76unluntktbzlp6qpb7sfgkal6vhfbe/
```

3.7 DNSLink: dominio real

Por defecto, al desplegar una aplicación web estática en IPFS, se le asignará un identificador único de cliente (CID). Sin embargo, los CID son largos, difíciles de recordar y poco intuitivos. Para eso vamos a asociar un dominio real.

DNSLink es un registro TXT en el DNS del dominio que apunta a un path de IPFS. Al apuntarlo a la clave IPNS (y no a un CID), el registro DNS se configura una sola vez: las actualizaciones del sitio se siguen haciendo con ipfs name publish sin tocar el DNS.

Como no es necesario comprar un dominio, se utiliza deSEC, un proveedor de DNS gratuito que otorga subdominios bajo dedyn.io con control total de la zona, lo que permite crear el registro con el prefijo _dnslink que exige la especificación de DNSLink.

1. Crear una cuenta en desec.io, confirmar el mail y registrar un dominio bajo dedyn.io (ej. maxiborrajo.dedyn.io).

2. En la gestión de registros del dominio, crear el registro TXT:

```
Type: TXT
Subname: _dnslink
TTL: 3600
Content:
"dnslink=/ipns/k51qzi5uqu5dlk7vmp29jh0f3v5zqvc39xyufg0ykul9oevmi2tb641rxm
cn1o"
```

El registro queda en _dnslink.maxiborrajo.dedyn.io, que es donde los resolvers de DNSLink lo buscan.

3. Una vez propagado el DNS (en deSEC suele ser cuestión de segundos o minutos), verificar el registro:

```
dig +short TXT _dnslink.maxiborrajo.dedyn.io
> "dnslink=/ipns/k51qzi5uqu5dlk7vmp29jh0f3v5zqvc39xyufg0ykul9oevmi2tb641rxmcn1o"
```

4. Verificar la resolución completa desde Kubo:

```
ipfs name resolve /ipns/maxiborrajo.dedyn.io
> /ipfs/bafybeiaotxdyji4bwktyq53jyfb76unluntktbzlp6qpb7sfgkal6vhfbe
```

5. Acceder al sitio por el dominio. Cualquier usuario de la red puede hacerlo desde su propio nodo:

```
ipfs cat /ipns/maxiborrajo.dedyn.io/index.html
```

O desde un navegador vía gateway (notar el namespace /ipns/, no /ipfs/):

```
http://127.0.0.1:8080/ipns/maxiborrajo.dedyn.io/
https://ipfs.io/ipns/maxiborrajo.dedyn.io/
```

Los usuarios con la extensión IPFS Companion pueden ingresar directamente a https://misitio.mooo.com: la extensión detecta el DNSLink y carga el contenido desde IPFS. Para que el dominio a secas funcione en un navegador sin extensión, hace falta además un registro A/CNAME apuntando a un gateway que soporte DNSLink. Pero esto también pierde las ventajas de IPFS, como la recuperación peer to peer, la resistencia a la censura, etc.

4. Problemas encontrados y soluciones

1. block was not found locally (offline)

   Error:

```
admin1@Deb:~$ ipfs cat
/ipfs/bafybeie5nqv6kd3qnfjupgvz34woh3oksc3iau6abmyajn7qvtf6d2ho34/quick-s
tart
Error: block was not found locally (offline): ipld: could not find
QmdncfsVm2h5Kqq9hPmU7oAVX2zTSVP3L869tgTbPYnsha
```

   Descripción: el bloque solicitado no está en el repositorio local y el nodo no está conectado a la red para buscarlo en otros peers (el daemon no está corriendo).

   Solución: iniciar el daemon y reintentar.

```
ipfs daemon
```

2. tar: Unexpected EOF in archive

   Error:

```
admin1@PruebaTp:~$ tar -xvzf kubo_v0.42.0_linux-amd64.tar.gz
gzip: stdin: unexpected end of file
tar: Unexpected EOF in archive
tar: Error is not recoverable: exiting now
```

   Descripción: el archivo comprimido está incompleto o corrupto; la descarga se interrumpió antes de terminar.

   Solución: eliminar el archivo y volver a descargarlo completo con wget antes de descomprimir.

3. vboxuser is not in the sudoers file

   Error:

```
vboxuser@TpPrueba2:~/kubo$ sudo bash install.sh
[sudo] password for vboxuser:
vboxuser is not in the sudoers file.
```

   Descripción: el usuario de la VM no tiene permisos de sudo, necesarios para que el script de instalación mueva el binario a /usr/local/bin.

   Solución: elevar privilegios con su, ejecutar la instalación como root y volver al usuario normal. La inicialización y uso del nodo (ipfs init, ipfs daemon) se hacen con el usuario normal, no con root.

```
su
bash install.sh
> Moved ./ipfs to /usr/local/bin
su vboxuser
```

4. Confusión entre namespaces /ipfs/ y /ipns/

   Error:

```
curl
https://ipfs.io/ipfs/k51qzi5uqu5dlk7vmp29jh0f3v5zqvc39xyufg0ykul9oevmi2tb
641rxmcn1o
> failed to resolve: could not choose a decoder: no decoder registered
for
> multicodec code 114 (0x72)
```

Descripción: las claves k51... no son CIDs de contenido sino nombres IPNS (multicodec 0x72 = libp2p-key). Al usarlas bajo /ipfs/, el gateway intenta decodificar el nombre como contenido y falla.

Solución: usar el namespace correcto según el tipo de identificador: /ipfs/ para CIDs inmutables (bafy..., Qm...) y /ipns/ para nombres mutables (k51..., DNSLink).

```
curl -L
https://ipfs.io/ipns/k51qzi5uqu5dlk7vmp29jh0f3v5zqvc39xyufg0ykul9oevmi2tb6
41rxmcn1o/
```

5. El gateway público encuentra el contenido pero no puede recuperarlo (nodo detrás de NAT)

   Error:

```
curl -L https://ipfs.io/ipns/k51qzi.../
> Unable to retrieve content within timeout period: timeout occurred after
> finding 3 provider(s) and connecting to 3 (phase: connecting to
providers)
```

   Descripción: la resolución IPNS funcionaba y el provider record estaba publicado en la DHT (el gateway encontraba nuestro peer ID), pero fallaba la transferencia de bloques por Bitswap: el nodo corría en una VM con red NAT de VirtualBox sumada al NAT del router, quedando inalcanzable para conexiones entrantes.

   Solución: cambiar la red de la VM a adaptador puente, eliminando la capa de NAT de VirtualBox. Con esto la recuperación pasó a depender solo del hole punching a través del NAT del router, que junto con el cache del gateway permitió el acceso vía gateways públicos.

6. FreeDNS no permite crear el registro _dnslink

   Error: Creation of records beginning with '_' are presently restricted to the domain owner only by default

   Como alternativa se creó el TXT directamente en el subdominio (sin prefijo). El registro propagó y dig lo veía, pero Kubo no lo resolvía:

```
dig +short TXT maxiborrajo.mooo.com
>"dnslink=/ipns/k51qzi5uqu5dlk7vmp29jh0f3v5zqvc39xyufg0ykul9oevmi2tb641rx
mcn1o"

ipfs name resolve /ipns/maxiborrajo.mooo.com
> Error: DNSLink lookup for "maxiborrajo.mooo.com." failed: could not
resolve
> name: DNSLink lookup could not find a TXT record
```

Descripción: FreeDNS restringe la creación de registros con prefijo _ en dominios compartidos al dueño del dominio. La alternativa del TXT en el dominio a secas no funciona porque la especificación de DNSLink (dnslink.dev) define el registro en _dnslink.<dominio>; el formato sin prefijo está deprecado y las versiones actuales de Kubo ya no lo consultan. dig lo encuentra porque consulta exactamente lo que se le pide; Kubo consulta únicamente _dnslink.<dominio>.

Solución: migrar a un proveedor DNS con control total de la zona que permita crear registros con prefijo _. Se usó deSEC (desec.io) con un dominio gratuito bajo dedyn.io.

5. Referencias

- Documentación oficial de instalación: https://docs.ipfs.tech/install/command-line/
- Conceptos IPFS: https://docs.ipfs.tech/concepts/
- IPNS: https://docs.ipfs.tech/concepts/ipns/
- NAT and Port Forwarding: https://docs.ipfs.tech/how-to/nat-configuration/#configuration-options
- DNSLink: https://docs.ipfs.tech/concepts/dnslink/#publish-content-path
- Repositorio Kubo: https://github.com/ipfs/kubo
- Verificador de disponibilidad: https://check.ipfs.network/
- deSEC (DNS gratuito): https://desec.io/
- Redirección de gateways públicos a inbrowser.link: https://docs.ipfs.tech/how-to/replace-public-gateways-with-self-hosted-ipfs/

6. Uso de IA

Se utilizó el modelo Fable 5 de Anthropic como asistente durante el desarrollo del trabajo.

Se utilizó en la resolución de errores durante la ejecución del software y para redacción de este mismo trabajo práctico

1. **Diagnóstico de problemas de red:** ante los timeouts intermitentes de los gateways públicos, la IA ayudó a interpretar la causa: al correr el nodo IPFS en una máquina virtual detrás de NAT, el nodo queda inalcanzable para conexiones entrantes, por lo que los gateways lo encuentran en la DHT pero no logran recuperar el contenido. Se pasó de red NAT a adaptador puente para reducir una capa de NAT, y se entendió que la recuperación restante dependía de hole punching (intermitente) y del caché de los gateways.
2. **Opciones de DNS gratuito:** permitió conocer alternativas de DNS gratuito para configurar DNSLink, y entender por qué FreeDNS no era viable (restricción de registros con prefijo _) y deSEC sí.
3. **Redacción:** asistencia en la redacción de la sección de problemas y soluciones a partir de las salidas reales de terminal obtenidas durante las pruebas.

Todos los comandos, salidas y capturas del trabajo corresponden a ejecuciones reales. La IA se usó para diagnóstico y redacción, no para generar resultados.