



|keep coding

# Módulo: Configurar una red de BESU

Caso práctico

**Víctor García Delgado**

## Contenido

1. Enunciado práctico .....	2
2. Desarrollo .....	3
2.1 Red Test.....	3
2.2 Red Productiva.....	7
Paso 1. Crear Directorios .....	7
Paso 2. Crear un archivo .....	8
Paso 3. Generar claves de nodo y un archivo .....	8
Paso 4. Copiar el archivo génesis al directorio .....	10
Paso 5. Copiar las claves privadas del nodo a los directorios.....	11
Paso 6. Inicie el primer nodo como nodo de arranque .....	12
Paso 7 Iniciar el Nodo-2, Nodo-3 y Nodo-4 .....	13
Paso 10 Confirmamos que la red privada está funcionando.....	16
3. Conclusiones.....	18

# 1. Enunciado práctico

BlockchainTech está empezando a utilizar Hyperledger Besu para implementar una red Blockchain para la trazabilidad de activos alimentarios de sus clientes. Para empezar, quieren desplegar una red de desarrollo con un único nodo y luego expandirla a una red de producción con 5 nodos, desplegando un contrato inteligente para comprobar su funcionalidad. Además, quieren llevar a cabo una mínima administración de la red mediante alguna herramienta de monitorización como Prometheus.

Después de varias semanas de espera, la empresa os ha contratado a vosotros para llevar a cabo el proyecto. ¡Enhorabuena!

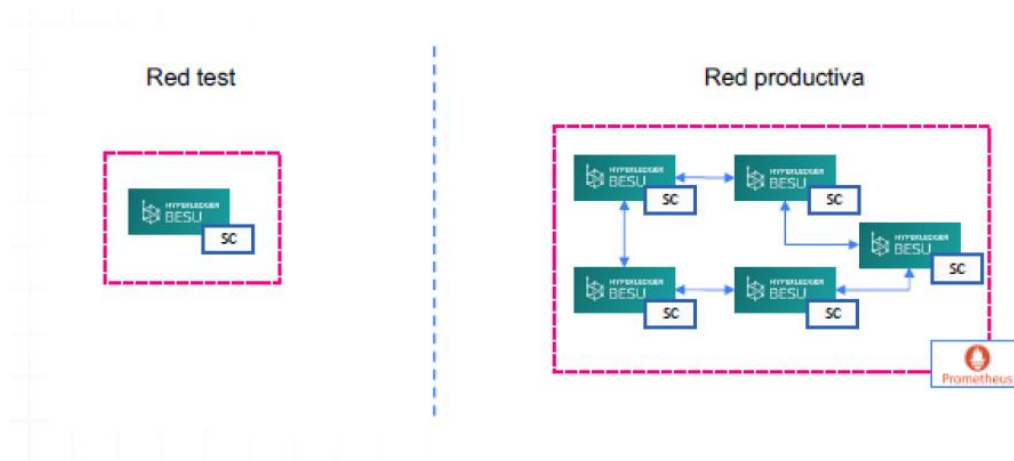
Durante el desarrollo del módulo utilizaremos un repositorio de código donde aparecen diferentes ejemplos de cara al despliegue de redes Hyperledger Besu llamado quorum dev quickstart. Para llevar a cabo la práctica, podéis utilizarlo como código base:

<https://github.com/Consensys/quorum-dev-quickstart>

La idea es que os facilite la generación de los scripts necesarios de despliegue de las redes solicitadas por el cliente.

Como resultado de la práctica, será necesario presentar los scripts de configuración y despliegue, un Smart Contract de ejemplo y un fichero README donde expliquéis el resultado de la práctica, justificando las decisiones que se han tomado.

Tanto la forma del despliegue de las redes como el despliegue del Smart Contract podéis realizarlas de la forma que más os convenga. Lo principal será explicar los motivos por los cuales se ha hecho de esa manera.



## 2. Desarrollo

### 2.1 Red Test

En primer lugar, nos hacemos un fork de quorum-dev-quickstart en nuestra ruta de git.

```
https://github.com/vgaciad/KeepcodingBESU
```

Posteriormente, nos traemos el repositorio a local para poder ejecutar el proyecto

```
git clone https://github.com/vgaciad/KeepcodingBESU
```

Ahora, una vez tenemos el repo en local, procedemos a desplegar la red de test de BESU apoyándonos en la herramienta *quorum-dev-quickstart*

```
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/KeepcodingBESU$ ls
files  index.js  LICENSE  npm-shrinkwrap.json  package.json  README.md  src  templates  tsconfig.json
```

Verificamos que cumplimos los prerequisites:

- [Docker and Docker-compose](#) v2 or higher -> ok
- On Windows, please use WSL2 kernels 5.15x or higher -> ¿¿ Para la práctica se utiliza una MB
- You can use either Docker Desktop or docker-engine (with the compose plugin) within the WSL2 environment -> ok
- [Nodejs](#) or [Yarn](#) -> ok, para node, pero nos falta npm (\* Lo instalamos sin problemas)

```

Procesando disparadores para man-db (2.10.2-1) ...
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/KeepcodingBESU$ source ~/.bashrc
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/KeepcodingBESU$ npm --version
9.2.0
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/KeepcodingBESU$ ^C

```

Ejecutamos el comando

```
npx quorum-dev-quickstart
```

Y vemos que levanta correctamente

```

KeepcodingBesu/ quorum-dev-quickstart/
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu$ npx quorum-dev-quickstart
Need to install the following packages:
  quorum-dev-quickstart@0.2.4
Ok to proceed? (y) y

  quorum
  developer
  quickstart

Welcome to the Quorum Developer Quickstart utility. This tool can be used
to rapidly generate local Quorum blockchain networks for development purposes
using tools like GoQuorum, Besu, and Tessera.

To get started, be sure that you have both Docker and Docker Compose
installed, then answer the following questions.

Which Ethereum client would you like to run? Default: [1]
  1. Hyperledger Besu
  2. GoQuorum

```

Ahora para probar, vamos a seleccionar las siguientes opciones.

1. Para seleccionar Hyperledger Besu
2. No (porque ya no hay soporte de tessera)
3. Loki (el más ligero)
4. No (Chainlens)

Verificamos que la instalación se ha realizado correctamente -> ok

```

Do you wish to enable support for monitoring your network with Blockscout? [N/y]
n
Where should we create the config files for this network? Please
choose either an empty directory, or a path to a new directory that does
not yet exist. Default: ./quorum-test-network

[✓] Installation complete.

To start your test network, run 'run.sh' in the directory, './quorum-test-network'
For more information on the test network, see 'README.md' in the directory, './quorum-test-network'
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu$

```

```

ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network$ ls
attach.sh  config  docker-compose.yml  filebeat  logstash  metricbeat  quorum-explorer  remove.sh  resume.sh  smart_contracts  static
chainlens  dapps  extra              list.sh   loki      promtail   README.md        restart.sh  run.sh         splunk          stop.sh
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network$

```

Posteriormente, realizamos un

```
./run.sh
```

Nos aparece un error de Docker-compose (pese a tenerlo instalado) y no levanta.

```

ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network$ ./run.sh
This script requires Docker compose but it's not installed.
Refer to documentation to fulfill requirements.
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network$ ls
attach.sh  config  docker-compose.yml  filebeat  logstash  metricbeat  quorum-explorer  remove.sh  resume.sh  smart_contracts  static
chainlens  dapps  extra              list.sh   loki      promtail   README.md        restart.sh  run.sh         splunk          stop.sh
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network$ docker-
compose --version
docker-compose version 1.29.2, build unknown
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quor

```

Vamos a probar de otra forma, pero si ejecutamos con el siguiente comando, levantando directamente los contenedores:

```

attach.sh  config  docker-compose.yml  filebeat  logstash  metricbeat  quorum-explorer  remove.sh  resume.sh  smart_contracts  static
chainlens  dapps  extra              list.sh   loki      promtail   README.md        restart.sh  run.sh         splunk          stop.sh
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network$ docker-compose -f docker-compose.yml up -d

```

```
docker-compose -f docker-compose.yml up -d
```

Parece que la red se ha levantado/creado correctamente

```
7a5c4dc1f629: Pull complete
Digest: sha256:67f60160f2d65f37f0a457109bfb5bd456ec630b7c29a399cd3ef6ae80048469
Status: Downloaded newer image for grafana/promtail:2.8.4
Creating quorum-test-network_promtail_1 ...
Creating quorum-test-network_prometheus_1 ...
Creating quorum-test-network_validator1_1 ...
Creating quorum-test-network_grafana_1 ...
Creating quorum-test-network_promtail_1 ... done
Creating quorum-test-network_prometheus_1 ... done
Creating quorum-test-network_validator1_1 ... done
ERROR: for quorum-test-network_grafana_1 Cannot start service grafana: driver failed programming external connectivity on endpoint
Creating quorum-test-network_loki_1 ... done
Creating quorum-test-network_validator3_1 ... done
Creating rpcnode ... done
Creating quorum-test-network_validator4_1 ... done
Creating quorum-test-network_validator2_1 ... done
Creating quorum-test-network_ethsignerProxy_1 ... done
Creating quorum-test-network_explorer_1 ... done

ERROR: for grafana Cannot start service grafana: driver failed programming external connectivity on endpoint
2): Bind for 0.0.0.0:3000 failed: port is already allocated
ERROR: Encountered errors while bringing up the project.
ubuntu@ubuntu-VirtualBox: ~/Keepcoding/Besu/quorum-test-network$
```

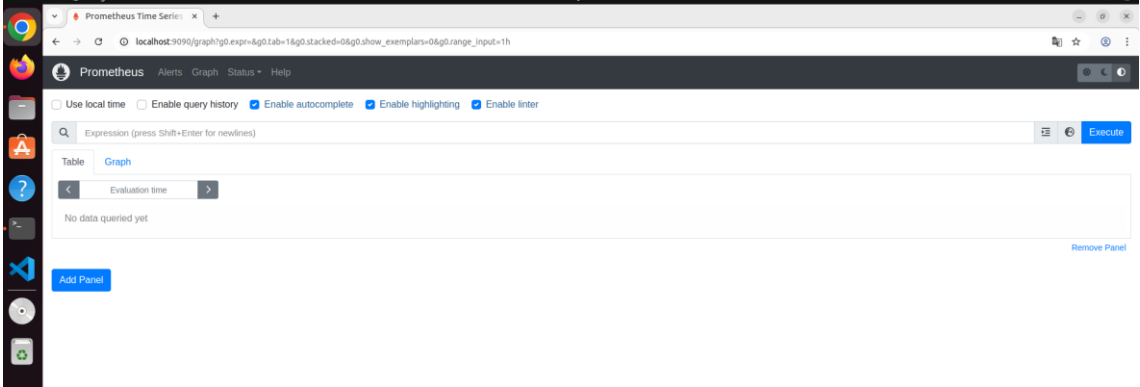
(\*) Aparece un error de grafana

Y verificamos también revisando los contenedores están correctamente levantados -> ok

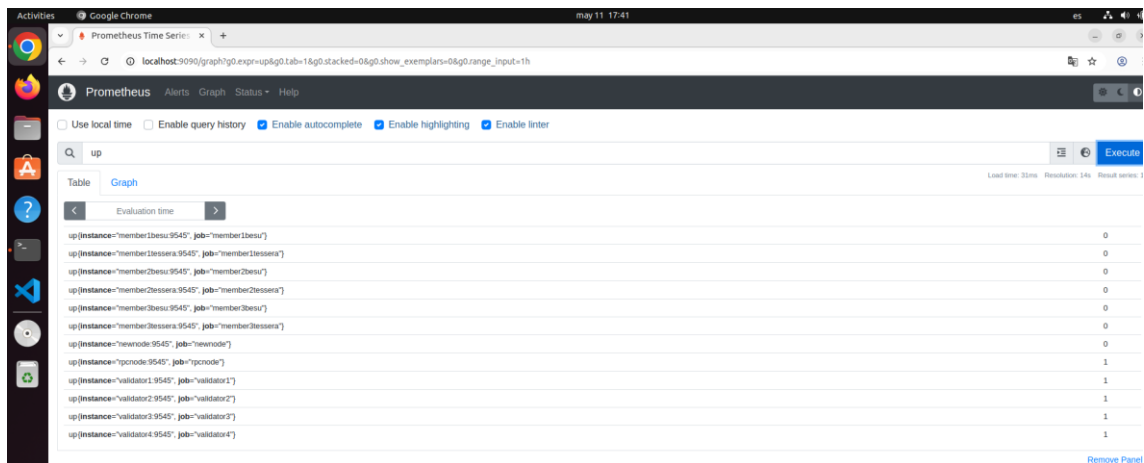
```
Docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
66a1a45d8a85   consensus/quorum-explorer:4f60191  "docker-entrypoint.s..." About a minute ago Up About a minute 0.0.0.0:25000->25000/tcp, :::25000->25000/tcp
742b10957ead   consensus/ethsigner:22.1.3         "/opt/ethsigner/bin/..." About a minute ago Up About a minute 0.0.0.0:18545->8545/tcp, :::18545->8545/tcp
be2e4aac72c3   hyperledger/besu:23.4.1            "/bin/bash -c 'nproc ..." About a minute ago Up About a minute (healthy) 8546-8547/tcp, 8550-8551/tcp, 0.0.0.0:21002->8545/tcp, :::21002->8545/tcp
cp, 0.0.0.0:32773->8545/tcp, :::32773->8545/tcp, 0.0.0.0:32770->30303/tcp, :::32770->30303/tcp quorum-test-network_validator2_1
b062defa0fa   hyperledger/besu:23.4.1            "/bin/bash -c 'nproc ..." About a minute ago Up About a minute (healthy) 8546-8547/tcp, 8550-8551/tcp, 0.0.0.0:21004->8545/tcp, :::21004->8545/tcp
cp, 0.0.0.0:32775->8545/tcp, :::32775->8545/tcp, 0.0.0.0:32772->30303/tcp, :::32772->30303/tcp quorum-test-network_validator4_1
e9ae32e30825   hyperledger/besu:23.4.1            "/bin/bash -c 'nproc ..." About a minute ago Up About a minute (healthy) 8546-8547/tcp, 8550-8551/tcp, 0.0.0.0:21003->8545/tcp, :::21003->8545/tcp
cp, 0.0.0.0:32774->8545/tcp, :::32774->8545/tcp, 0.0.0.0:32771->30303/tcp, :::32771->30303/tcp quorum-test-network_validator3_1
b06c493baf0a   hyperledger/besu:23.4.1            "/bin/bash -c 'nproc ..." About a minute ago Up About a minute (healthy) 8547/tcp, 8550-8551/tcp, 0.0.0.0:8545-8546->8545-8546/tcp, :::8545-8546->8545-8546/tcp
->8545-8546/tcp, 30303/tcp rpcnode
6c2522f6536   grafana/loki:2.8.4                 "/usr/bin/loki -conf..." About a minute ago Up About a minute 0.0.0.0:3100->3100/tcp, :::3100->3100/tcp
c3432787270a   hyperledger/besu:23.4.1            "/bin/bash -c 'nproc ..." About a minute ago Up About a minute (healthy) 8546-8547/tcp, 8550-8551/tcp, 0.0.0.0:21001->8545/tcp, :::21001->8545/tcp
cp, 0.0.0.0:32769->8545/tcp, :::32769->8545/tcp, 0.0.0.0:32768->30303/tcp, :::32768->30303/tcp quorum-test-network_validator1_1
9999dc42f2b9   prom/prometheus:v2.46.0            "/bin/prometheus ..." About a minute ago Up About a minute 0.0.0.0:9090->9090/tcp, :::9090->9090/tcp
fdb0c9c24d8   grafana/promtail:2.8.4              "/usr/bin/promtail ..." About a minute ago Up About a minute 0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
301c1d9a6cb   grafana/grafana:8.3.4               "/run.sh"                11 days ago Up 56 minutes 0.0.0.0:9100->9100/tcp, :::9100->9100/tcp
714e0f6b774a   prom/node-exporter:v1.3.1           "/bin/node_exporter ..." 11 days ago Up 56 minutes node-exporter
3a16bdf2f598   google/cadvisor:latest              "/usr/bin/cadvisor ..." 11 days ago Restarting (255) 40 seconds ago cadvisor
```

Verificamos que podemos acceder a prometheus -> ok

<http://localhost:9090/>



Realizamos alguna búsqueda (up) y verificamos correctamente como podemos ejecutar queries.



Paramos la red:

```
ubuntu@ubuntu-VirtualBox: ~/Keepcoding/Besu/quorum-test-network$ docker-compose down
Stopping quorum-test-network_explorer_1 ... done
Stopping quorum-test-network_ethsignerProxy_1 ... done
Stopping quorum-test-network_validator2_1 ... done
Stopping quorum-test-network_validator4_1 ... done
Stopping quorum-test-network_validator3_1 ... done
Stopping rpcnode ... done
Stopping quorum-test-network_loki_1 ... done
Stopping quorum-test-network_validator1_1 ... done
Stopping quorum-test-network_prometheus_1 ... done
Stopping quorum-test-network_promtail_1 ... done
Removing quorum-test-network_explorer_1 ... done
Removing quorum-test-network_ethsignerProxy_1 ... done
Removing quorum-test-network_validator2_1 ... done
Removing quorum-test-network_validator4_1 ... done
Removing quorum-test-network_validator3_1 ... done
Removing rpcnode ... done
Removing quorum-test-network_loki_1 ... done
Removing quorum-test-network_validator1_1 ... done
Removing quorum-test-network_prometheus_1 ... done
Removing quorum-test-network_grafana_1 ... done
Removing quorum-test-network_promtail_1 ... done
Removing network quorum-dev-quickstart
```

## 2.2 Red Productiva

Para realizar una red productiva, nos basaremos en la documentación

<https://besu.hyperledger.org/private-networks/tutorials/qbft>

### Paso 1. Crear Directorios

En primer lugar, creamos la estructura de directorios:

```
QBFT-Network/
├── Node-1
│   └── data
```



```

├── Node-2
│   ├── data
│   ├── Node-3
│   ├── data
└── Node-4
    ├── data

```

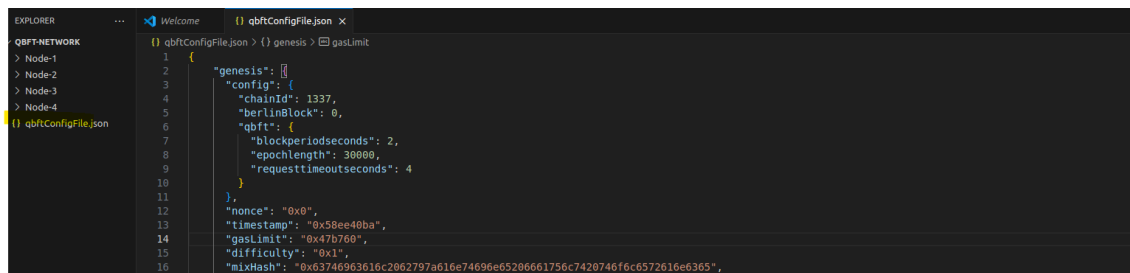
```

ubuntu@ubuntu-virtualbox: ~/Keepcoding/Besu$ cd QBFT-Network/
ubuntu@ubuntu-virtualbox:~/Keepcoding/Besu/QBFT-Network$ ls
Node-1 Node-2 Node-3 Node-4
ubuntu@ubuntu-virtualbox:~/Keepcoding/Besu/QBFT-Network$

```

## Paso 2. Crear un archivo

Posteriormente creamos el archivo `qbftConfigFile.json` y lo guardamos en el QBFT-Network directorio -> ok



```

EXPLORER
  QBFT-NETWORK
    > Node-1
    > Node-2
    > Node-3
    > Node-4
    qbftConfigFile.json
  qbftConfigFile.json
  Welcome
  qbftConfigFile.json > {} genesis > gasLimit
  1 {
  2   "genesis": {},
  3   "config": {
  4     "chainId": 1337,
  5     "berlinBlock": 0,
  6     "qbft": {
  7       "blockperiodseconds": 2,
  8       "epochlength": 30000,
  9       "requesttimeoutseconds": 4
  10    },
  11   },
  12   "nonce": "0x0",
  13   "timestamp": "0x58ee40ba",
  14   "gasLimit": "0x47b760",
  15   "difficulty": "0x1",
  16   "mixHash": "0x63746963616c2062797a616e74696e65206661756c742074666c6572616e6365",

```

## Paso 3. Generar claves de nodo y un archivo

Nos ubicamos en el directorio y ejecutamos el comando:

```

besu operator generate-blockchain-config --config-file=qbftConfigFile.json --
to=networkFiles --private-key-file-name=key

```

Para que nos funcione, ejecutamos el siguiente comando:

```

docker run --rm \
-v $(pwd):/app \
hyperledger/besu:latest \
operator generate-blockchain-config \
--config-file=/app/qbftConfigFile.json \
--to=/app/networkFiles \
--private-key-file-name=key

```

Este comando docker run es la forma de ejecutar besu operator generate-blockchain-config ... en tu entorno, dado que Besu no está instalado directamente.

### ¿Qué hace este comando y qué debes esperar?

- Se descargará la imagen más reciente de hyperledger/besu (si no la tienes ya).

- Se iniciará un contenedor temporal basado en esa imagen (--rm).
- Tu directorio actual (QBFT-Network) se hará accesible dentro del contenedor como /app.
- Dentro del contenedor, se ejecutará el comando besu operator generate-blockchain-config.
- Este comando leerá /app/qbftConfigFile.json (que es tu qbftConfigFile.json en el directorio QBFT-Network).
- Generará los archivos de configuración de la blockchain (como genesis.json), las claves de los nodos (key es el prefijo para los archivos de clave privada) y otros archivos necesarios.
- Guardará toda esta salida dentro de un directorio llamado networkFiles, que se creará *dentro de tu directorio actual* (QBFT-Network) gracias al volumen montado (-v \$(pwd):/app).

La primera vez que lo ejecutamos, nos aparece el siguiente error:

```
992e2f70b83c: Pull complete
4f4fb700ef54: Pull complete
dfc3659fade9: Pull complete
Digest: sha256:00468488df234ead7c0e450ffcd5bc1ac3ed62f458a0de42ba029155ecd5331
Status: Downloaded newer image for hyperledger/besu:latest
java.lang.IllegalArgumentException: Output directory already exists.
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/QBFT-Network$ ls
```

Procedemos a depurarlo.

En primer lugar, eliminamos la carpeta networkFiles que ha dado conflicto y volvemos a ejecutar el comando con la opción **--force-overwrite** en el comando generate-blockchain-config. Esto le dice a Besu que si el directorio de salida ya existe, lo sobrescriba. (\* Ayuda proporcionada por Gemini)

```
docker run --rm \
-v $(pwd):/app \
hyperledger/besu:latest \
operator generate-blockchain-config \
--config-file=/app/qbftConfigFile.json \
--to=/app/networkFiles \
--private-key-file-name=key \
--force-overwrite
```

Después de varias pruebas, conseguimos ejecutarlo con el siguiente comando:

```
docker run --rm \
-v $(pwd):/app \
hyperledger/besu:latest \
```

```
/bin/bash -c "ls -la /app && besu operator generate-blockchain-config --config-file=/app/qbftConfigFile.json --to=/app/networkFiles --private-key-file-name=key"
```

Dónde obtenemos como salida lo siguiente:

```
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/QBFT-Network$ docker run --rm \
-v $(pwd):/app \
--entrypoint /bin/bash \
hyperledger/besu:latest \
-c "ls -la /app && besu operator generate-blockchain-config --config-file=/app/qbftConfigFile.json --to=/app/networkFiles --private-key-file-name=key"
total 28
drwxr-xr-x 6 besu besu 4096 May 11 16:51 .
drwxr-xr-x 1 root root 4096 May 11 16:57 ..
drwxr-xr-x 3 besu besu 4096 May 11 16:38 Node-1
drwxr-xr-x 3 besu besu 4096 May 11 16:38 Node-2
drwxr-xr-x 3 besu besu 4096 May 11 16:38 Node-3
drwxr-xr-x 3 besu besu 4096 May 11 16:38 Node-4
-rw-rw-r-- 1 besu besu 1593 May 11 16:33 qbftConfigFile.json
2025-05-11 16:57:36.102+00:00 | main | INFO | GenerateBlockchainConfig | Generating 4 nodes keys.
2025-05-11 16:57:36.111+00:00 | main | INFO | GenerateBlockchainConfig | Generating keypair for node 0.
2025-05-11 16:57:36.243+00:00 | main | INFO | GenerateBlockchainConfig | Generating keypair for node 1.
2025-05-11 16:57:36.262+00:00 | main | INFO | GenerateBlockchainConfig | Generating keypair for node 2.
2025-05-11 16:57:36.286+00:00 | main | INFO | GenerateBlockchainConfig | Generating keypair for node 3.
2025-05-11 16:57:36.298+00:00 | main | INFO | GenerateBlockchainConfig | Generating QBFT extra data.
2025-05-11 16:57:36.312+00:00 | main | INFO | GenerateBlockchainConfig | Writing genesis file.
```

Tiene buena pinta

Esto nos genera lo siguiente:

```
File Edit Selection View Go Run Terminal Help QBFT-Network
EXPLORER
- QBFT-NETWORK
  - networkFiles
    - genesis.json
  - keys
  - Node-1/data
  - Node-2/data
  - Node-3/data
  - Node-4/data
  - qbftConfigFile.json
networkFiles/genesis.json
1 {
2   "config": {
3     "chainId": 1337,
4     "berlinBlock": 0,
5     "qbft": {
6       "blockperiodseconds": 2,
7       "epochlength": 3000,
8       "requesttimeoutseconds": 4
9     }
10  },
11  "nonce": "0x0",
12  "timestamp": "0x0bee40ba",
13  "gasLimit": "0x47b76d",
14  "difficulty": "0x1",
15  "mixHash": "0x6374696316c2862797a616e74696e65286661756c742074666572816e6305",
16  "coinbase": "0x000000000000000000000000000000000000000000000000",
17  "alloc": {
18    "fe3b557e8fb62b99f91b6721be55ceb828dbd73": {
19      "privateKey": "8f2a5594938a9610f5fb23b5883af3b4ecb3c3bb792cbcefb1542c692be63",
20      "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
21      "balance": "0xad78ebc5ac62000000"
22    },
23    "0273866990aba346e1408e9345bc60c78a8BEf57": {
24      "privateKey": "c87589a1c867bde78be793e6fa76538b6382a4c6241e54e9ec9a8f44dc8d3",
25      "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored",
26      "balance": "0x00000000000000000000000000000000"
27    },
28    "f17f52151eBEFC7334FAD088c5784D77216b732": {
29      "privateKey": "a6a8e5ccbf04598405997ee2d52d26330761378875853c36d94e974d182f",
30      "comment": "private key and this comment are ignored. In a real chain, the private key should NOT be stored"
31    }
32  }
33 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
-rw-rw-r-- 1 besu besu 1593 May 11 16:33 qbftConfigFile.json
2025-05-11 16:57:36.102+00:00 | main | INFO | GenerateBlockchainConfig | Generating 4 nodes keys.
```

- genesis.json- El archivo génesis incluye la extraData propiedad que especifica que los cuatro nodos son validadores.
- Un directorio para cada nodo nombrado utilizando la dirección del nodo y que contiene la clave pública y privada para cada nodo.

Paso 4. Copiar el archivo génesis al directorio

Copiar el genesis.json archivo al QBFT-Network directorio -> ok

```
cp networkFiles/genesis.json .
```

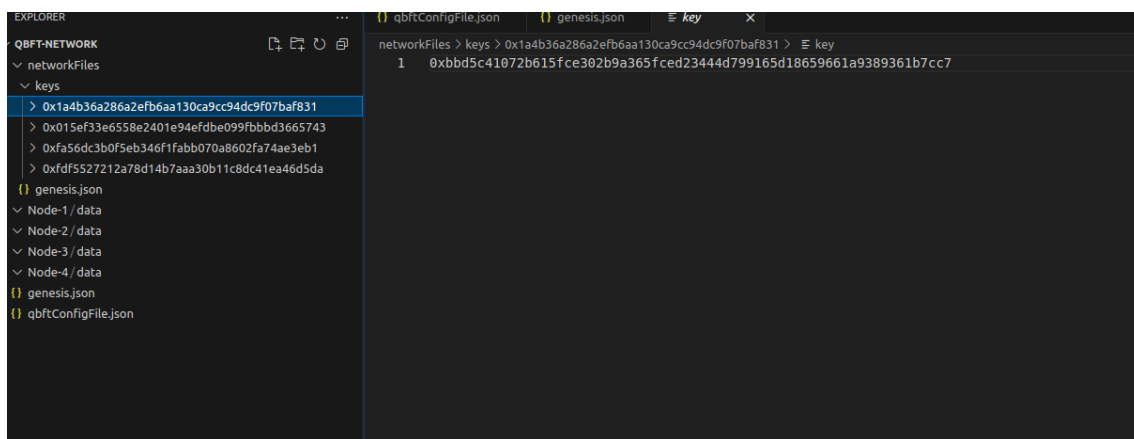
```
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/QBFT-Network$ ls
genesis.json  networkFiles  Node-1  Node-2  Node-3  Node-4  qbftConfigFile.json
```

## Paso 5. Copiar las claves privadas del nodo a los directorios

Para ello:

```
QBFT-Network/  
├── genesis.json  
├── Node-1  
│   ├── data  
│   │   ├── key  
│   │   └── key.pub  
├── Node-2  
│   ├── data  
│   │   ├── key  
│   │   └── key.pub  
├── Node-3  
│   ├── data  
│   │   ├── key  
│   │   └── key.pub  
├── Node-4  
│   ├── data  
│   │   ├── key  
│   │   └── key.pub
```

Debemos coger las claves privadas:



y distribuir las claves privadas y públicas generadas en networkFiles a los directorios data de cada nodo (Node-1 a Node-4), renombrándolas en el proceso para que simplemente se llamen key (privada) y key.pub (pública) dentro de cada carpeta data.

Nos ubicamos en nuestra ~/Keepcoding/Besu/QBFT-Network

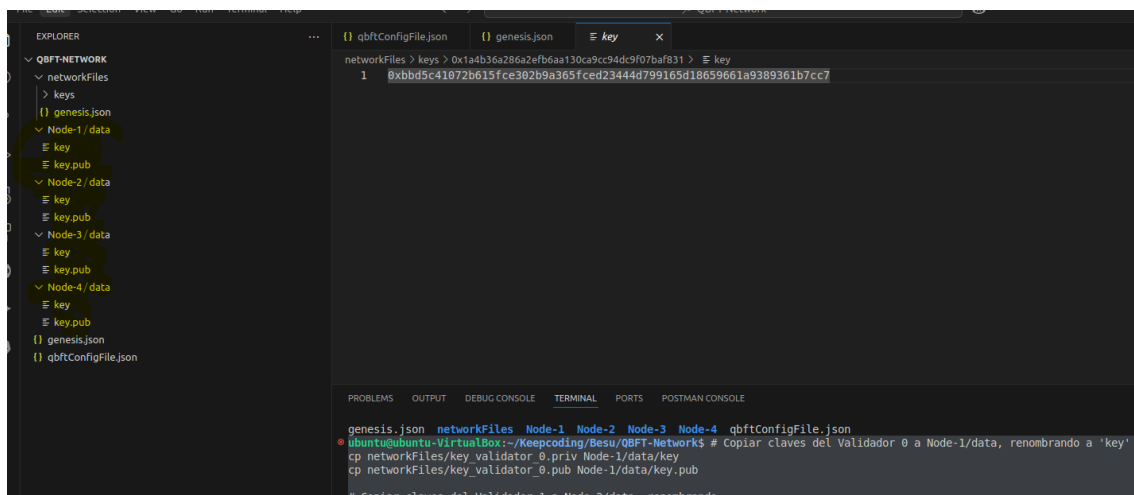
```
# Copiar claves del Validador 0 a Node-1/data, renombrando a 'key' y 'key.pub'  
cp networkFiles/key_validator_0.priv Node-1/data/key  
cp networkFiles/key_validator_0.pub Node-1/data/key.pub
```

```
# Copiar claves del Validador 1 a Node-2/data, renombrando
cp networkFiles/key_validator_1.priv Node-2/data/key
cp networkFiles/key_validator_1.pub Node-2/data/key.pub

# Copiar claves del Validador 2 a Node-3/data, renombrando
cp networkFiles/key_validator_2.priv Node-3/data/key
cp networkFiles/key_validator_2.pub Node-3/data/key.pub

# Copiar claves del Validador 3 a Node-4/data, renombrando
cp networkFiles/key_validator_3.priv Node-4/data/key
cp networkFiles/key_validator_3.pub Node-4/data/key.pub
```

Lo hacemos a mano, enviando los archivos de la carpeta **networkfiles** a los diferentes nodos. Se nos quedan los ficheros de la siguiente forma:



## Paso 6. Inicie el primer nodo como nodo de arranque

```
besu --data-path=data --genesis-file=./genesis.json --rpc-http-enabled --rpc-http-
api=ETH,NET,QBFT --host-allowlist="*" --rpc-http-cors-origins="all" --
profile=ENTERPRISE
```

(\* ) Podríamos optar por instalar besu, pero aunque los comandos sean más largos, vamos a aprovechar las ventajas de dockers aislando el entorno de Besu. Así que de momento, no necesitamos instalar Java o Besu directamente en nuestro sistema base, fácil de levantar/bajar y limpiar los contenedores (Ayuda de IA Gemini, pero prefiero seguir decantándome por esta opción)

Procedemos ejecutando el siguiente comando (cruzando dedos):

```
docker run -d --name node-1 \
-v $(pwd)/Node-1/data:/app/data \
```

```
-v $(pwd)/genesis.json:/app/genesis.json \
-w /app \
-p 8545:8545 \
-p 30303:30303 \
hyperledger/besu:latest \
--data-path=data \
--genesis-file=genesis.json \
--rpc-http-enabled \
--rpc-http-api=ETH,NET,QBFT \
--host-allowlist="*" \
--rpc-http-cors-origins="all" \
--profile=ENTERPRISE
```

La salida tiene muy buena pinta, parece que el nodo se ha levantado correctamente:

```
--genesis-file=genesis.json \
--rpc-http-enabled \
--rpc-http-api=ETH,NET,QBFT \
--host-allowlist="*" \
--rpc-http-cors-origins="all" \
--profile=ENTERPRISE
5763de7899ae2d98881b7932beeca478d0acef9e856814e76427c03d3444c79c
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/QBFT-Network$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5763de7899ae	hyperledger/besu:latest	"besu-entry.sh --dat..."	5 seconds ago	Up 4 seconds (health: starting)	8546-8547/tcp, 0.0.0.0:8545->8545/tcp, :::8545->8545/tcp, 0.0.0.0:30303->30303/tcp, :::30303->30303/tcp, 8550-8551/tcp	node-1
d01c1d9ae6cb	grafana/grafana:8.3.4	"/run.sh"	11 days ago	Up 3 hours	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	grafana
f14e0f8bf74a	prom/node-exporter:v1.3.1	"/bin/node_exporter ..."	11 days ago	Up 3 hours	0.0.0.0:9100->9100/tcp, :::9100->9100/tcp	node-exporter
9a16bdf2f598	google/cadvisor:latest	"/usr/bin/cadvisor -..."	11 days ago	Restarting (255) 56 seconds ago		cadvisor

```
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/QBFT-Network$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5763de7899ae	hyperledger/besu:latest	"besu-entry.sh --dat..."	5 seconds ago	Up 4 seconds (health: starting)	8546-8547/tcp, 0.0.0.0:8545->8545/tcp, :::8545->8545/tcp, 0.0.0.0:30303->30303/tcp, :::30303->30303/tcp, 8550-8551/tcp	node-1
d01c1d9ae6cb	grafana/grafana:8.3.4	"/run.sh"	11 days ago	Up 3 hours	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	grafana
f14e0f8bf74a	prom/node-exporter:v1.3.1	"/bin/node_exporter ..."	11 days ago	Up 3 hours	0.0.0.0:9100->9100/tcp, :::9100->9100/tcp	node-exporter
9a16bdf2f598	google/cadvisor:latest	"/usr/bin/cadvisor -..."	11 days ago	Restarting (255) 56 seconds ago		cadvisor

## Paso 7 Iniciar el Nodo-2, Nodo-3 y Nodo-4

Inicie otra terminal, cambie al Node-2 directorio e inicie Node-2 especificando la URL enode del Node-1 copiada al iniciar Node-1 como nodo de arranque:

Nodo-2

```
docker run -d --name node-2 \
```

```

-v $(pwd)/Node-2/data:/app/data \
-v $(pwd)/genesis.json:/app/genesis.json \
-w /app \
-p 8546:8546 \
-p 30304:30304 \
hyperledger/besu:latest \
--data-path=data \
--genesis-file=genesis.json \
--
bootnodes=enode://66b131c68722b32f4568a4a0a48b4d470bd1c197d9b569bc5af0
cd0f04199fea93a28290f0c238d66c83239d4ef19e2f455a5d1d2ac50e5dce5a51e2eeb
4cab6@127.0.0.1:30303 \
--p2p-port=30304 \
--rpc-http-enabled \
--rpc-http-api=ETH,NET,QBFT \
--host-allowlist="*" \
--rpc-http-cors-origins="all" \
--rpc-http-port=8546 \
--profile=ENTERPRISE \
--node-private-key-file=/app/data/key

```

OK

```

--profile=ENTERPRISE \
--node-private-key-file=/app/data/key
6c219c16c42c60ff6558395f83426fd9f45de680b9d777efb821d6dc6e2021bf
ubuntu@ubuntu-VirtualBox:~/Keppcoding/Besu/QBFT-Network$

```

Node-3 ->OK

```

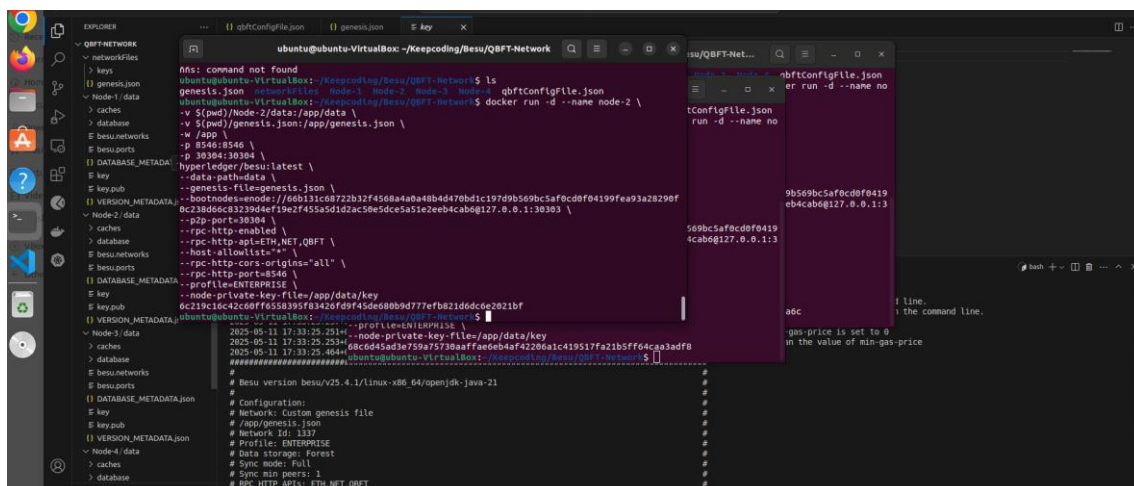
docker run -d --name node-3 \
-v $(pwd)/Node-3/data:/app/data \
-v $(pwd)/genesis.json:/app/genesis.json \
-w /app \
-p 8547:8547 \
-p 30305:30305 \
hyperledger/besu:latest \
--data-path=data \
--genesis-file=genesis.json \
--
bootnodes=enode://66b131c68722b32f4568a4a0a48b4d470bd1c197d9b569bc5af0
cd0f04199fea93a28290f0c238d66c83239d4ef19e2f455a5d1d2ac50e5dce5a51e2eeb
4cab6@127.0.0.1:30303 \
--p2p-port=30305 \
--rpc-http-enabled \
--rpc-http-api=ETH,NET,QBFT \
--host-allowlist="*" \
--rpc-http-cors-origins="all" \

```

```
--rpc-http-port=8547 \  
--profile=ENTERPRISE \  
--node-private-key-file=/app/data/key
```

Node-4 -> OK

```
docker run -d --name node-4 \  
-v $(pwd)/Node-4/data:/app/data \  
-v $(pwd)/genesis.json:/app/genesis.json \  
-w /app \  
-p 8548:8548 \  
-p 30306:30306 \  
hyperledger/besu:latest \  
--data-path=data \  
--genesis-file=genesis.json \  
--  
bootnodes=enode://66b131c68722b32f4568a4a0a48b4d470bd1c197d9b569bc5af0  
cd0f04199fea93a28290f0c238d66c83239d4ef19e2f455a5d1d2ac50e5dce5a51e2eeb  
4cab6@127.0.0.1:30303 \  
--p2p-port=30306 \  
--rpc-http-enabled \  
--rpc-http-api=ETH,NET,QBFT \  
--host-allowlist="" \  
--rpc-http-cors-origins="all" \  
--rpc-http-port=8548 \  
--profile=ENTERPRISE \  
--node-private-key-file=/app/data/key
```



```
ubuntu@ubuntu-VirtualBox: ~/Keepcoding/Besu/QBFT-Network  
$ docker run -d --name node-4 -v $(pwd)/Node-4/data:/app/data -v $(pwd)/genesis.json:/app/genesis.json -w /app -p 8548:8548 -p 30306:30306 hyperledger/besu:latest --data-path=data --genesis-file=genesis.json --bootnodes=enode://66b131c68722b32f4568a4a0a48b4d470bd1c197d9b569bc5af0cd0f04199fea93a28290f0c238d66c83239d4ef19e2f455a5d1d2ac50e5dce5a51e2eeb4cab6@127.0.0.1:30303 --p2p-port=30306 --rpc-http-enabled --rpc-http-api=ETH,NET,QBFT --host-allowlist="" --rpc-http-cors-origins="all" --rpc-http-port=8548 --profile=ENTERPRISE --node-private-key-file=/app/data/key  
$ docker ps  
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS  
node-4              hyperledger/besu   --data-path=data --ge...   2025-05-11 17:33:25   Up 2 minutes        0.0.0.0:30306->30306, 0.0.0.0:8548->8548  
$ docker logs node-4  
Besu version besu/v25.4.1/linux-x86_64/openjdk-java-21  
#  
# Configuration:  
# Network: Custom genesis file  
# App: genesis.json  
# Network Id: 1337  
# Profile: ENTERPRISE  
# Data storage: Forest  
# Sync mode: Full  
# Sync min peers: 1  
# RPC HTTP APIs: ETH,NET,QBFT  
# RPC HTTP port: 8548  
#  
# The command line.  
gas-price is set to 0 in the value of min-gas-price
```

Verificamos con un

```
Docker ps  
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/QBFT-Network$ docker ps
```



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS				
NAMES				
a63ca0f19d88	hyperledger/besu:latest	"besu-entry.sh --dat..."	About a minute ago	Up About a minute (healthy)
8545-8547/tcp, 8550-8551/tcp, 0.0.0.0:8548->8548/tcp, :::8548->8548/tcp, 30303/tcp, 0.0.0.0:30306->30306/tcp, :::30306->30306/tcp node-4				
68c6d45ad3e7	hyperledger/besu:latest	"besu-entry.sh --dat..."	2 minutes ago	Up 2 minutes (healthy)
8545-8546/tcp, 8550-8551/tcp, 0.0.0.0:8547->8547/tcp, :::8547->8547/tcp, 30303/tcp, 0.0.0.0:30305->30305/tcp, :::30305->30305/tcp node-3				
6c219c16c42c	hyperledger/besu:latest	"besu-entry.sh --dat..."	3 minutes ago	Up 3 minutes (healthy)
8545/tcp, 8547/tcp, 8550-8551/tcp, 0.0.0.0:8546->8546/tcp, :::8546->8546/tcp, 30303/tcp, 0.0.0.0:30304->30304/tcp, :::30304->30304/tcp node-2				
5763de7899ae	hyperledger/besu:latest	"besu-entry.sh --dat..."	16 minutes ago	Up 16 minutes (healthy)
8546-8547/tcp, 0.0.0.0:8545->8545/tcp, :::8545->8545/tcp, 0.0.0.0:30303->30303/tcp, :::30303->30303/tcp, 8550-8551/tcp node-1				
d01c1d9ae6cb	grafana/grafana:8.3.4	"/run.sh"	11 days ago	Up 3 hours
0.0.0.0:3000->3000/tcp, :::3000->3000/tcp grafana				
f14e0f8bf74a	prom/node-exporter:v1.3.1	"/bin/node_exporter ..."	11 days ago	Up 3 hours
0.0.0.0:9100->9100/tcp, :::9100->9100/tcp node-exporter				
9a16bdf2f598	google/cadvisor:latest	"/usr/bin/cadvisor -..."	11 days ago	Restarting (255) 33 seconds ago

Ahora que ya parece que tenemos la red operativa vamos a probar:

## Paso 10 Confirmamos que la red privada está funcionando

Inicie otra terminal, use curl para llamar al [qbft\\_getvalidatorsbyblocknumber](#) método API JSON-RPC y confirme que la red tiene cuatro validadores:

```
curl -X POST --data
'{"jsonrpc": "2.0", "method": "qbft_getValidatorsByBlockNumber", "params": ["latest"], "id": 1}' localhost:8545/ -H "Content-Type: application/json"
```

Ok, verificamos que la red funciona correctamente

```

root@kali:~/hyperledger/besu# curl -X POST --data '{"jsonrpc": "2.0", "method": "qbft_getValidatorsByBlockNumber", "params": ["latest"], "id": 1}' localhost:8545/ -H "Content-Type: application/json"
{"jsonrpc": "2.0", "id": 1, "result": [{"validator": "0x015e73e055be2401e04efdb099fbbd3665743", "enclave": "0x015e73e055be2401e04efdb099fbbd3665743"}, {"validator": "0x015e73e055be2401e04efdb099fbbd3665743", "enclave": "0x015e73e055be2401e04efdb099fbbd3665743"}, {"validator": "0x015e73e055be2401e04efdb099fbbd3665743", "enclave": "0x015e73e055be2401e04efdb099fbbd3665743"}, {"validator": "0x015e73e055be2401e04efdb099fbbd3665743", "enclave": "0x015e73e055be2401e04efdb099fbbd3665743"}]}

```

Vamos a ver si podemos desplegar un Smart contract, tras varias pruebas para poder instalar hardat, probamos a ver si sale bien

```
npx hardhat run scripts/deploy.js --network localhost
```

```
ubuntu@ubuntu-VirtualBox:~/Keepcoding/Besu/quorum-test-network/smart_contracts$ npx hardhat run scripts/deploy.js --network localhost
Error HH1: You are not inside a Hardhat project.
```

For more info go to <https://hardhat.org/HH1> or run Hardhat with --show-stack-traces

Parece que seguimos teniendo algún tipo de error con las dependencias de hardhat.

(\*) Dado que el próximo módulo es para el despliegue de Smart Contract sobre fabric, y debido al tiempo disponible, dejamos aquí pendiente el despliegue del Smart contract sobre la red de Besu.

No obstante, comprobamos como la red está ejecutando bloques y hay algo que tampoco entendemos que es:

```
docker logs node-1 -f
```

```
2025-05-11 17:33:41.660+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 2 which will expire in 16 seconds
2025-05-11 17:33:57.672+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 3 which will expire in 32 seconds
2025-05-11 17:34:29.674+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 4 which will expire in 64 seconds
2025-05-11 17:35:33.676+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 5 which will expire in 128 seconds
2025-05-11 17:37:41.679+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 6 which will expire in 256 seconds
2025-05-11 17:41:57.681+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 7 which will expire in 512 seconds
2025-05-11 17:50:29.683+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 8 which will expire in 1024 seconds
2025-05-11 18:07:33.699+00:00 | BftProcessorExecutor-QBFT-0 | INFO | RoundTimer | Moved to round 9 which will expire in 2048 seconds
```

Porque no vemos los mensajes cruciales como Imported block #X ... o Produced block #Y ... después de la fase inicial de inicio. Esto, combinado con el hecho de que las rondas están progresando (lo que suele ocurrir si el consenso se atasca esperando votos o propuestas), sugiere que los nodos no están logrando alcanzar el quorum necesario para sellar y añadir nuevos bloques a la cadena.

Por lo que parece que los nodos **no se están encontrando y conectando correctamente entre sí** a través de la red P2P. Aunque están intentando el consenso (por eso ves las rondas), si no hay conexión entre ellos, no pueden enviarse las votaciones o propuestas de bloques necesarias para completarlo.

A continuación, habría que verificar la conectividad entre los nodos. Pese a casi tenerlo, por adversidades, no disponemos de más tiempo para poder seguir depurando antes de poder realizar la entrega.

### 3. Conclusiones

Con la realización de esta práctica, hemos aprendido dos partes bien diferenciadas.

En la primera parte (**Red Test**):

1. El quorum-dev-quickstart es una herramienta muy útil para **acelerar la fase inicial** de configuración de una red de desarrollo Besu.
2. Su función principal es **generar automáticamente** los archivos necesarios para la red, como el docker-compose.yml, scripts para controlar los nodos y, a menudo, un proyecto de Smart Contracts de ejemplo listo para usar (como un proyecto Hardhat).
3. Simplifica considerablemente los pasos iniciales de configuración manual, proporcionando una base rápida para empezar a trabajar con una red local.
4. Aunque agiliza la generación de archivos, el uso posterior de los scripts o proyectos generados puede requerir familiaridad con Docker Compose, Node.js/NPM y la estructura específica que crea la herramienta.

En resumen, es una excelente herramienta para obtener rápidamente los archivos base para tu red de desarrollo, permitiéndote pasar más tiempo trabajando con los nodos y Smart Contracts en lugar de tener que configurarlos desde cero.

En la segunda parte (**Red Productiva**) hemos aprendido conceptos como:

1. Hyperledger Besu se puede ejecutar dentro de contenedores Docker.
2. Utilizamos comandos docker run para ejecutar utilidades de Besu (como la generación de configuración), manejando mapeo de volúmenes y permisos.
3. Logramos generar y distribuir correctamente los archivos esenciales como el genesis.json y las claves de los nodos.
4. Pudimos iniciar cada nodo Besu en su propio contenedor Docker, configurando sus puertos, datos y la conexión al nodo de arranque.
5. Confirmamos la accesibilidad RPC de los nodos y que la red reconoce a sus validadores.
6. Aprendimos a usar docker logs -f para monitorear la actividad de los nodos en tiempo real.
7. Observamos que, aunque los nodos estaban intentando el consenso, no se veían mensajes de producción de bloques, lo que sugirió un posible problema de conectividad P2P.
8. Intentamos desplegar un Smart Contracts usando un proyecto Hardhat, pero no conseguimos resolver las dependencias

Pese a no haber conseguido con éxito realizar todas las partes, la práctica ha sido un recorrido muy completo para aprender los pasos fundamentales para levantar, configurar y verificar una red privada de Besu en tu máquina virtual.