1. Despliega la red. Parece que existen algunos contenedores que no se levantan correctamente. Identifícalos y resuelve los problemas asociados.

Ejecutar los comandos para instalar los binarios de fabric

```
curl -sSLO
https://raw.githubusercontent.com/hyperledger/fabric/main/scri
pts/install-fabric.sh && chmod +x install-fabric.sh
./install-fabric.sh b
```

Levantamos la red de fabric

```
./network.sh up createChannel -ca
```

En la línea 41 en la parte de los orderes estaba mal escrita la palabra Hyperledger

```
orderer.example.com:
  container_name: orderer.example.com
  image: hyperledger/fabric-orderer:latest
  labels:
    service: hyperledger-fabric
  environment:
    - FABRIC_LOGGING_SPEC=INFO
    - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
    - ORDERER_GENERAL_LISTENPORT=7050
    - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
    - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
    # enabled TLS
    - ORDERER_GENERAL_TLS_ENABLED=true
    - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
    - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
    - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
    - ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE=/var/hyperledger/orderer/tls/server.crt
    - ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY=/var/hyperledger/orderer/tls/server.key
    - ORDERER_GENERAL_CLUSTER_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
    - ORDERER_GENERAL_BOOTSTRAPMETHOD=none
    - ORDERER_CHANNELPARTICIPATION_ENABLED=true
    - ORDERER_ADMIN_TLS_ENABLED=true
    - ORDERER_ADMIN_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
    - ORDERER_ADMIN_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
    - ORDERER_ADMIN_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
    - ORDERER_ADMIN_TLS_CLIENTROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
    - ORDERER_ADMIN_LISTENADDRESS=0.0.0.0:7053
    - ORDERER_OPERATIONS_LISTENADDRESS=orderer.example.com:9443
    - ORDERER_METRICS_PROVIDER=prometheus
```

En la línea 90 en la parte de los peer estaba mal escrita la palabra server.

```
peer0.org1.example.com:
  container_name: peer0.org1.example.com
  image: hyperledger/fabric-peer:latest
  labels:
    service: hyperledger-fabric
  environment:
    - FABRIC_CFG_PATH=/etc/hyperledger/peercfg
    - FABRIC_LOGGING_SPEC=INFO
    #- FABRIC_LOGGING_SPEC=DEBUG
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_PROFILE_ENABLED=false
    - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
    - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
    - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    # Peer specific variables
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:7051
    - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:7052
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/fabric/msp
    - CORE_OPERATIONS_LISTENADDRESS=peer0.org1.example.com:9444
    - CORE_METRICS_PROVIDER=prometheus
    - CHAINCODE_AS_A_SERVICE_BUILDER_CONFIG={"peername":"peer0org1"}
    - CORE_CHAINCODE_EXECUTETIMEOUT=300s
  volumes:
    - ../organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com:/etc/hyperledger/fabric
    - peer0.org1.example.com:/var/hyperledger/production
  working_dir: /root
```

2. Despliega el chaincode “merca-chaincode”. Aunque el proceso de ciclo de vida del chaincode es satisfactorio, parece que este no se despliega correctamente. Identifica el/los problemas y resuélvelos. Documenta este proceso de investigación y resolución.

Instalamos y desplegamos el chaincode

```
./network.sh deployCC -ccn merca-chaincode -ccl javascript
-ccv 1.0.0 -ccp ../merca-chaincode
```

Problema en el chaincode que esta ubicado en la carpeta merca-chaincode/lib/assetTrasnfer.js, le falta el cierre de una llave {} en la línea 89.

```
70      }
71
72      // CreateAsset issues a new asset to the world state with given details.
73      async CreateAsset(ctx, id, color, size, owner, appraisedValue) {
74          const exists = await this.AssetExists(ctx, id);
75          if (exists) {
76              throw new Error(`The asset ${id} already exists`);
77          }
78
79          const asset = {
80              ID: id,
81              Color: color,
82              Size: size,
83              Owner: owner,
84              AppraisedValue: appraisedValue,
85          };
86          // we insert data in alphabetic order using 'json-stringify-deterministic' and 'sort-keys-recursive'
87          await ctx.stub.putState(id, Buffer.from(stringify(sortKeysRecursive(asset))));
88          return JSON.stringify(asset);
89      }
90
91      // ReadAsset returns the asset stored in the world state with given id.
```

3. Desde Merca-link nos comentan que los logs de los peers no proporcionan la información deseada. Modifica el logging de la red para que se ajuste a los siguientes requerimientos:
- Los logs de los orderers deben estar en formato JSON.
- Los logs de los peers deben de mostrar la fecha y la hora al final.
- Dado que estamos en fase de pruebas, los logs de los peers deben de tener un nivel de logging de DEBUG por defecto. Además, los logs asociados al logger “gossip” tendrán un nivel por defecto de WARNING y los del logger de “chaincode” de INFO para no sobrecargar demasiado los logs.

Lanzamos las variables de entorno para modificar el contenido:

```
FABRIC_LOGGING_FORMAT=json
```

```
FABRIC_LOGGING_FORMAT="%{message} %{color}%{time:2006-01-02 15:04:05.000 MST}
[%{module}] %{shortfunc} -> %{level:.4s}
%{id:03x}%{color:reset}"
```

```
FABRIC_LOGGING_SPEC=warning:msp,gossip=info:chaincode=info
```

4. Los técnicos de Merca-link nos muestran su preocupación acerca de las claves criptográficas, que se almacenan en texto plano dentro de los directorios de la red. Como una primera medida de securización de las claves y de las operaciones criptográficas sugieren el acoplamiento de un HSM a las CAs, utilizándose softhsm2 para esta fase de pruebas y configurando un token diferente para cada CA.

Pues lo primero que hacer es instalar los prerrequisitos que nos piden, que lo tenemos en el repo de la clase en el punto 3.

Instrucciones para inicio nativo del servidor con HSM (ejecutando el binario directamente):

Prerrequisitos

- Instala softhsm2 con:

```
sudo apt install softhsm2
```

- Este SoftHSM necesita la librería libssl1.1. Instálala con:

```
sudo apt install -y libssl1.1
```

- Verificamos que tenemos la librería "libsofthsm2.so" en la ruta /usr/lib:

```
ls /usr/lib
```

- Creamos nuestro token con

```
softhsm2-util --init-token --slot 0 --label fabric
```

- Por defecto la configuración de SOFTHSM2 está en /etc/softhsm2.conf.
- Exporta la variable de entorno SOFTHSM2_CONF para indicar al programa dónde se encuentra la configuración:

```
export SOFTHSM2_CONF=/etc/softhsm2.conf
```

- Por defecto, los tokens se almacenan en /var/lib/softhsm2/tokens. Esta carpeta puede ser un tanto restrictiva a la hora de crear y consultar los tokens. Crea una nueva carpeta "tokens" donde almacenaremos los tokens del softhsm. Yo la crearé en mi directorio home:

```
cd /home/daniel mkdir tokens
```

- Modifica el fichero de configuración /etc/softhsm2.conf para indicarle dónde está la nueva carpeta que almacena los tokens: línea "directories.tokendir = /home/daniel/tokens"

Pasamos al punto 4 para desplegarlo en Docker y seguimos estos pasos:

Reconstrucción de la imagen

Las imágenes por defecto no dan soporte a pkcs#11, así que debemos de reconstruirlas. Te recomiendo que pares la test network o cualquier contenedor corriendo una fabric-ca y después hagas:

```
git clone https://github.com/hyperledger/fabric-ca.git # Clonamos repo de fabric-ca cd fabric-ca # Entramos a la carpeta clonada
```

Ubuntu:20.04 -> Edita el Dockerfile de la fabric-ca situado dentro de "fabric-ca/images/fabric-ca". Añade "RUN apt install -y libssl1.1" en la línea 53 justo debajo de "RUN pat update".

Reconstruye las imágenes con: (UBUNTU 20.04) > make docker GO_TAGS=pkcs11 (UBUNTU 22.04) > make docker GO_TAGS=pkcs11 UBUNTU_VER=22.04

Tras estos pasos comprobamos la nueva imagen con:

```
docker image ls
```

Ejecutamos estos comandos:

1.- Ejecutar "docker-compose up -d" desde esta carpeta. 2.- Ejecuta "docker ps" para ver si el contenedor de la CA está corriendo correctamente. 3.- Si no es así podrás ver el contenedor caído con "docker ps -a". 4.- En cualquier caso puedes ver los logs del contenedor con "docker logs <CONTAINER_ID> -f"

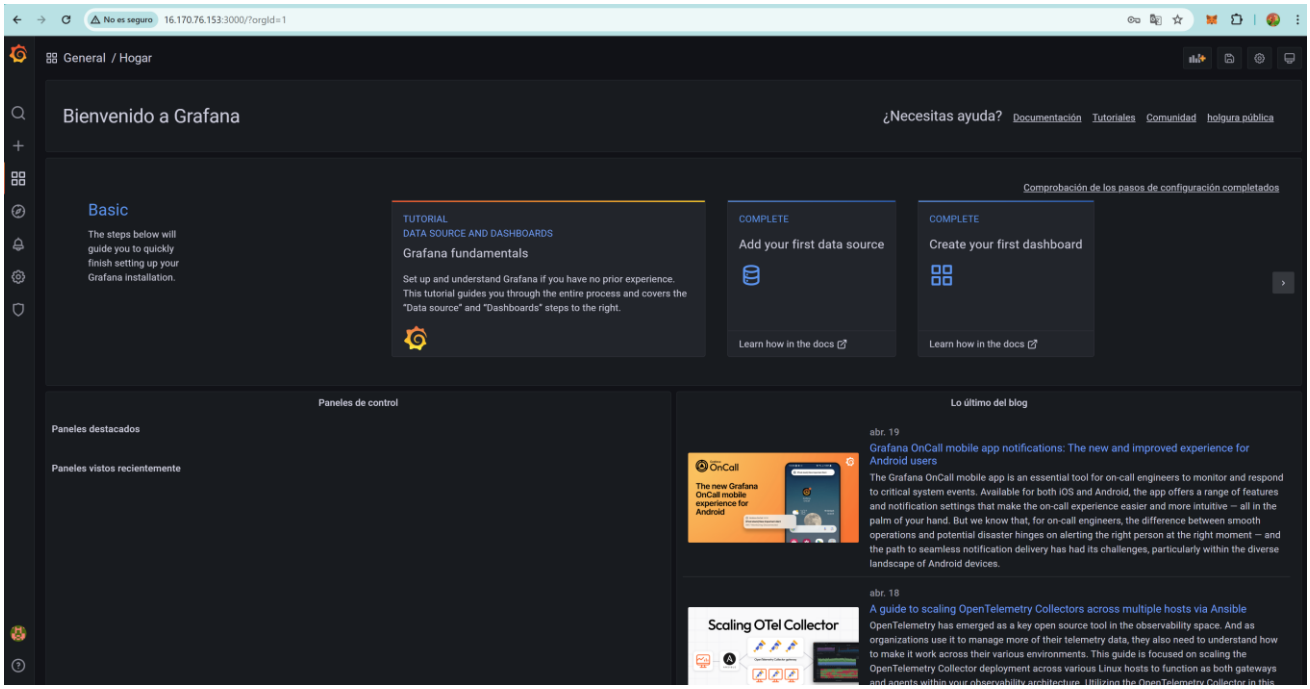
5. Así mismo, explícales cómo configurar y utilizar el fabric-ca-client con el softHSM configurado. Registra, a modo de prueba, un usuario de nombre “merca-admin” y contraseña “merka-12345”, que tenga capacidad para registrar usuarios clientes y nuevos peers.

6. Los merca-ingenieros de Merca-link consideran que tener herramientas de monitorización a su disposición agilizaría estas tareas de administración en futuras ocasiones. Despliega sendas instancias de Prometheus y Grafana para satisfacer estas necesidades.

Nos vamos a la carpeta merca-chain/prometeus-grafana y ejecutamos el comando:

Docker compose up

Y ahora accedemos a nuestra herramienta de monitoreo grafico a través de la ip de nuestra maquina y puerto.



7. Investiga sobre los despliegues distribuidos (esto es, en distintos nodos) de Hyperledger Fabric, y resume en unas pocas líneas tus recomendaciones y los principales puntos a tener en cuenta para que Merka-link despliegue su red de forma distribuida. No olvides referencias los artículos/blogs/páginas web que has consultado.

En resumen, es repartir nuestros peer y ordere a lo largo y ancho de nuestra red con diferente host.

- Cada host puede pertenecer a una organización específica según las reglas de negocio y políticas establecidas.
- La arquitectura y distribución de componentes y servicios en la red blockchain se definen en función de estas consideraciones

Aquí dejo referencia:

https://usuarioperu.com/2019/11/18/hyperledger-fabric-docker-swarm-multiples-hosts/