Hyperledger Fabric

Cómo desplegar una red de Fabric

Práctica:

Víctor García Delgado

Contenido

Introducción

Esta guía se detallan los pasos necesarios para desplegar una red Hyperledger Fabric basada en la test-network, ampliándola para incluir tres organizaciones:

1. Fabricante
2. Transportista
3. Distribuidor

También se intentará implementar un chaincode personalizado que gestiona el estado de los productos en la cadena de suministro. A lo largo de este documento, se explicará cómo configurar la red, desplegar el chaincode, probar su funcionalidad e incluir documentación en un archivo README.md.

A continuación, se muestran los pasos que realizaremos:

1. clonamos el repositorio de fabric-samples
2. configuramos la red test-network
3. implementamos el chaincode
4. Probamos la red

1. Clonamos el repositorio de fabric-samples

Para empezar, debemos hacer un fork del repositorio oficial de Fabric Samples y clonarlo en nuestro entorno de trabajo:

```
# Clonar el repositorio
git clone https://github.com/hyperledger/fabric-samples.git
cd fabric-samples
```

En primer lugar, vamos a verificar que la red se levanta correctamente inicialmente y tenemos los repositorios correctamente sincronizados

Ejecutamos

```
./network.sh up
```

Verificamos que la red funciona correctamente con Org1, Org2 y Orderer

```
configtx                                prometheus-grafana
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=v2.5.12
DOCKER_IMAGE_VERSION=v2.5.12
/home/ubuntu/Projects/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating orderer.example.com    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATED         STATUS                  PORTS
                                                                                           NAMES
3961c4fc3dfc   hyperledger/fabric-orderer:latest   "orderer"           2 seconds ago   Up Less than a second   0.0.0.0:7050->7050/tcp, :::7050->705
cp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
a657550efd1a   hyperledger/fabric-peer:latest      "peer node start"   2 seconds ago   Up Less than a second   0.0.0.0:7051->7051/tcp, :::7051->705
cp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                               peer0.org1.example.com
```

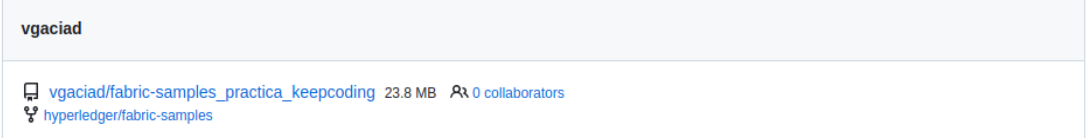
Ilustración 1. Test-Network levantada

Pero si intentamos configurar o dar de alta fabric en otro directorio, da error y no conseguimos levantar la red, por lo que vamos a mantener el repositorio proporcionado en la máquina virtual para la continuación de la práctica.

A continuación, verificamos que tenemos el fork correctamente creado del proyecto hyperledger para poder descargar y commitear los cambios realizados en git

https://github.com/hyperledger/fabric-samples

https://github.com/vgaciad/Hyperledger_final



A partir de aquí hacemos el:

```
git pull origin main
git push origin main
```

Para tener correctamente actualizada la rama de nuestro github dónde realizaremos los cambios requeridos.

Una vez mergeados los cambios, verificamos que la red sigue funcionando correctamente con un

```
./network.sh up
```

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'crypto
LOCAL_VERSION=v2.5.12
DOCKER_IMAGE_VERSION=v2.5.12
/home/ubuntu/Projects/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating orderer.example.com    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATED        STATUS                  PORTS
                                                                                          NAMES
9b80d6d1f06a   hyperledger/fabric-orderer:latest   "orderer"           1 second ago   Up Less than a second   0.0.0.0:7050->705
p, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
228735e9d2b5   hyperledger/fabric-peer:latest      "peer node start"   1 second ago   Up Less than a second   0.0.0.0:7051->705
p, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                              peer0.org1.example.com
37a84c1f7e5f   hyperledger/fabric-peer:latest      "peer node start"   1 second ago   Up Less than a second   0.0.0.0:9051->905
p, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                    peer0.org2.example.com
```

Ilustración 2. *Test-Network levantada*

Verificamos también que el comando, crea el canal correctamente -> ok

```
./network.sh up createChannel
```

```
+++ cat /home/ubuntu/Projects/fabric-samples/test-network/channel-artifacts/config_update.json
++ echo '{"payload":{"header":{"channel_header":{"channel_id":"mychannel", "type":2}},"data":{"config_update":{' '"cha
ated_data":' '{},' '"read_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Org2MSP":' '{' '"groups":'
olicies":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Endorsement":' '{' '"
null, '"version":' '"0"' '},' '"Readers":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Write
'"policy":' null, '"version":' '"0"' '}' '},' '"values":' '{' '"MSP":' '{' '"mod_policy":' '"",' '"value":' null, '"ve
n":' '"0"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '},' '"mod_poli
'"values":' '{},' '"version":' '"0"' '},' '"write_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Or
"mod_policy":' '"Admins",' '"policies":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"
d_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Readers":' '{' '"mod_policy":' '"",' '"policy":' null, '
":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '}' '},' '"values":' '{' '"AnchorPeers":' '{' '"mod
' '{' '"anchor_peers":' '[' '{' '"host":' '"peer0.org2.example.com",' '"port":' 9051 '}' ']' '},' '"version":' '"0"' '
 '"",' '"value":' null, '"version":' '"0"' '}' '},' '"version":' '"1"' '}' '},' '"mod_policy":' '"",' '"policies":' '{
:' '"0"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '}}}}'
++ configtxlator proto_encode --input /home/ubuntu/Projects/fabric-samples/test-network/channel-artifacts/config_updat
.Envelope --output /home/ubuntu/Projects/fabric-samples/test-network/channel-artifacts/Org2MSPanchors.tx
2025-03-30 12:30:51.726 CEST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-03-30 12:30:51.740 CEST 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ ./network.sh up createChannel^C
```

Ilustración 3. createChannel ok

Verificamos que podemos desplegar un chaincode sobre la red -> ok

```
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go
```

```
{
        "approvals": {
                "Org1MSP": true,
                "Org2MSP": true
        }
}
Checking the commit readiness of the chaincode definition successful on peer0.org2 on channel 'mychannel'
Using organization 1
Using organization 2
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/ubuntu/
es/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --channelID mychannel --name basic
alhost:7051 --tlsRootCertFiles /home/ubuntu/Projects/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/
mple.com-cert.pem --peerAddresses localhost:9051 --tlsRootCertFiles /home/ubuntu/Projects/fabric-samples/test-network/organizatio
org2.example.com/tlsca/tlsca.org2.example.com-cert.pem --version 1.0 --sequence 1
+ res=0
2025-03-30 12:43:53.473 CEST 0001 INFO [chaincodeCmd] ClientWait -> txid [3d320e29dc36af68aac4a42b6d5aa97dc036d981c774b70b57534f6
with status (VALID) at localhost:7051
2025-03-30 12:43:53.503 CEST 0002 INFO [chaincodeCmd] ClientWait -> txid [3d320e29dc36af68aac4a42b6d5aa97dc036d981c774b70b57534f6
with status (VALID) at localhost:9051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ doc
```

Ilustración 4. Chaincode go -> ok

Verificamos los Docker corriendo con (*No es posible la instalación de Docker-Desktop):

```
docker ps -a
```

```
                                COMMAND                  CREATED          STATUS          PORTS
                                                                         NAMES
48bf82dc446c   dev-peer0.org1.example.com-basic_1.0-6a6f743366675481339c7ddda5f281a441c9ad8c13e32a9a5c07b892e44de105-8dafef778
fbd66ebcf5c5961f724400d8ad29fe0   "chaincode -peer.add…"   2 minutes ago    Up 2 minutes
                                                                         dev-peer0.org1.example.com-basic_1.0-6a6f7433666754813
3e32a9a5c07b892e44de105
43cc0448c10f   dev-peer0.org2.example.com-basic_1.0-6a6f743366675481339c7ddda5f281a441c9ad8c13e32a9a5c07b892e44de105-40c3a7fdf
b387d562ff30140b3935c60c15aa16f   "chaincode -peer.add…"   2 minutes ago    Up 2 minutes
                                                                         dev-peer0.org2.example.com-basic_1.0-6a6f7433666754813
3e32a9a5c07b892e44de105
b15d71b0cf7b   hyperledger/fabric-peer:latest
                                "peer node start"        16 minutes ago   Up 16 minutes   0.0.0.0:7051->7051/tcp, :::7051->7
444/tcp, :::9444->9444/tcp                                               peer0.org1.example.com
d25701c1167b   hyperledger/fabric-peer:latest
                                "peer node start"        16 minutes ago   Up 16 minutes   0.0.0.0:9051->9051/tcp, :::9051->9
.0:9445->9445/tcp, :::9445->9445/tcp                                     peer0.org2.example.com
02b33f464812   hyperledger/fabric-orderer:latest
                                "orderer"                16 minutes ago   Up 16 minutes   0.0.0.0:7050->7050/tcp, :::7050->7
053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ docker ps -a
```

Ilustración 5. Docker levantados

(*) Por un tema de la KVM (Kernel-based Virtual Machine) no es posible instalar Docker-Desktop, aparece el siguiente error:

INFO: Your CPU does not support KVM extensions,

Y ahora enviamos una transacción a la red desde el peer1:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
```

```
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
```

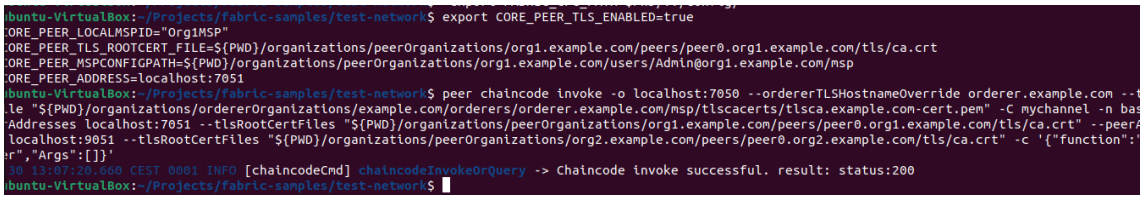


Ilustración 6. Transacción ok. Status 200

Verificamos que los asset se crean correctamente -> ok

```
peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}' | jq
```

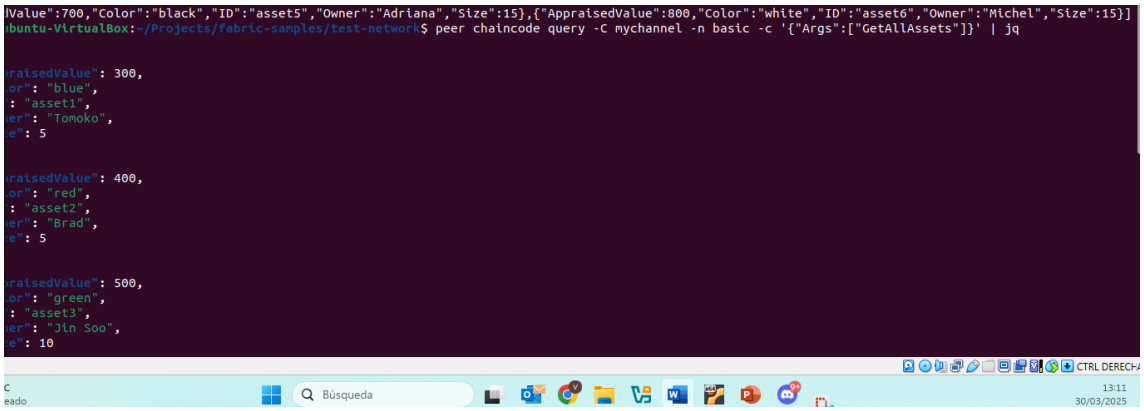


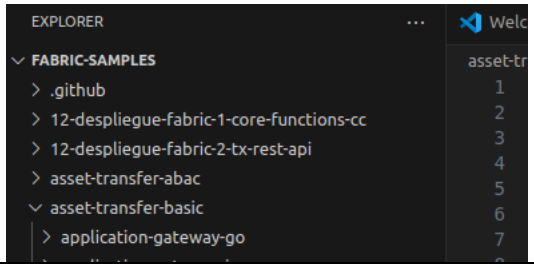
Ilustración 7. Los assets se crean correctamente

Tiramos la red y comenzamos las configuraciones

También nos traemos ambos repositorios proporcionados en clase para poder interactuar con la red:

```
git clone https://github.com/KeepCodingBlockchain-2/12-despliegue-fabric-2-tx-rest-api
```

git clone https://github.com/KeepCodingBlockchain-2/12-despliegue-fabric-2-tx-rest-api



A continuación, procedemos a realizar la configuración de la red.

2. Configuramos la red test-network

La red de prueba por defecto tiene dos organizaciones, pero para este caso de uso debemos agregar una tercera organización.

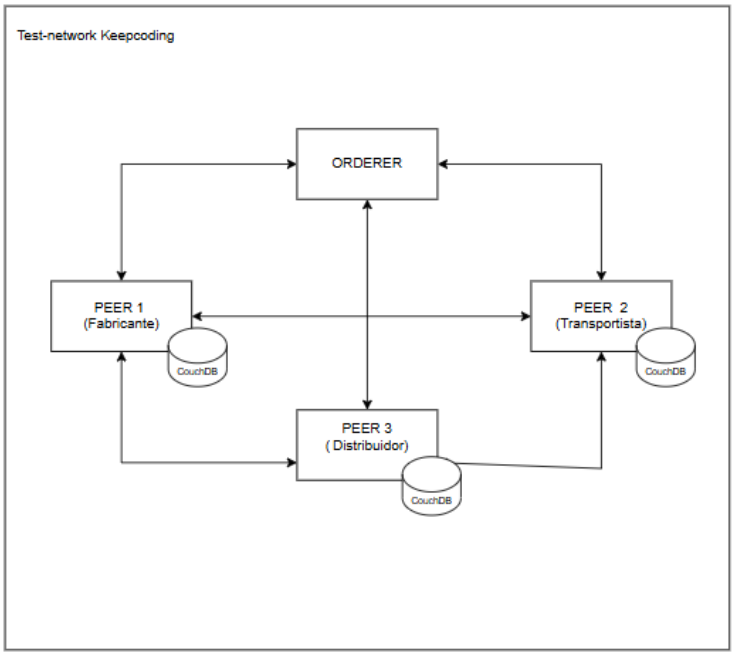


Ilustración 8. Configuración de red

2.1 Agregar la organización Distribuidor

A continuación, vamos agregar la nueva organización Distribuidor a la red test-network en Hyperledger Fabric.

1. Copiamos la configuración de una de las organizaciones existentes (Org1 o Org2) y realizamos las modificamos para la nueva Organización Distribuidor (Org3)

```
./network.sh up -s createChannel -s couchdb
```

Revisamos el fichero addOrg3.sh para revisar como se tiene que crear la organización3

Nos situamos en el directorio de la Org3 y ejecutamos

```
cd addOrg3
./addOrg3.sh up -s couchdb
```

Aparece el siguiente error:

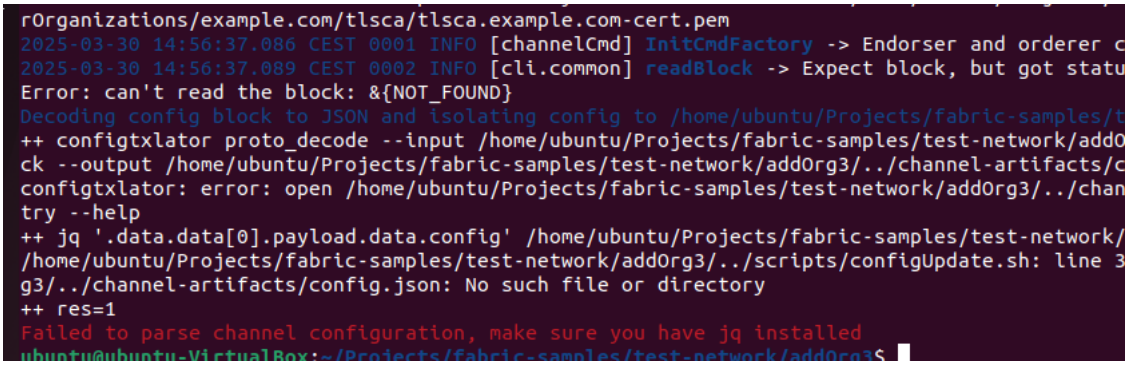


Ilustración 9. Error al intentar incluir la org3

Vamos a probar de nuevo repitiendo todo el proceso -> Org3 ok

Ilustración 10. Organización 3 añadida correctamente

Verificamos los contenedores dockers corriendo, vemos correctamente la org3 sobre coachDB -> ok

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network/addOrg3$ docker ps -a
CONTAINER ID   IMAGE                               COMMAND                  CREATED         STATUS         PORTS
                                                                                            NAMES
eb03a8e78751   hyperledger/fabric-peer:latest      "peer node start"        3 minutes ago   Up 3 minutes   7051/tcp, 0.0.0.0:11051->11051/tcp, :::1105
1->11051/tcp                                                                                peer0.org3.example.com
ed57f46f0028   couchdb:3.4.2                       "tini -- /docker-ent…"   3 minutes ago   Up 3 minutes   4369/tcp, 9100/tcp, 0.0.0.0:9984->5984/tcp,
 :::9984->5984/tcp                                                                          couchdb4
fe910ea99139   hyperledger/fabric-peer:latest      "peer node start"        4 minutes ago   Up 4 minutes   0.0.0.0:7051->7051/tcp, :::7051->7051/tcp,
0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                                   peer0.org1.example.com
837523798d56   hyperledger/fabric-peer:latest      "peer node start"        4 minutes ago   Up 4 minutes   0.0.0.0:9051->9051/tcp, :::9051->9051/tcp,
7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                         peer0.org2.example.com
c7b7b118e640   couchdb:3.4.2                       "tini -- /docker-ent…"   4 minutes ago   Up 4 minutes   4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp,
 :::7984->5984/tcp                                                                          couchdb1
08b917e6f323   couchdb:3.4.2                       "tini -- /docker-ent…"   4 minutes ago   Up 4 minutes   4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp,
 :::5984->5984/tcp                                                                          couchdb0
dc05e1f68f38   hyperledger/fabric-orderer:latest   "orderer"                4 minutes ago   Up 4 minutes   0.0.0.0:7050->7050/tcp, :::7050->7050/tcp,
0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network/addOrg3$
```

Ilustración 11. Contenedores levantados

Verificamos que podemos desplegar un chaincode:

```
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go
```

La Org1 y Org2 están ok, pero tenemos que configurar la Org3

```
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: false]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: false]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$
```

Ilustración 12. Org3MSP: False

Para ello, vamos a realizar la siguiente configuración a continuación

2. Editar los archivos de configuración en test-network para incluir la nueva organización.

Ejecutamos los siguientes comandos para poder desplegar los chaincodes sobre la Organización 3

```
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org3MSP"
export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt
export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp
export CORE_PEER_ADDRESS=localhost:11051
```

3. Implementar el chaincode

Creamos el paquete

```
peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-go/ --lang golang --label basic_1.0
```

Lo instalamos en el chaincode

```
peer lifecycle chaincode install basic.tar.gz
```

Verificamos el id:

```
peer lifecycle chaincode queryinstalled

ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: basic_1.0:6a6f743366675481339c7ddda5f281a441c9ad8c13e32a9a5c07b892e44de105, Label: basic_1.0
```

Y lo introducimos por comando:

```
export CC_PACKAGE_ID = 6a6f743366675481339c7ddda5f281a441c9ad8c13e32a9a5c07b892e44de105
```

Luego

```
peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" --channelID mychannel --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
peer lifecycle chaincode querycommitted --channelID mychannel --name basic --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
```

Le pasamos los datos del orderer

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverri
de orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example
.com-cert.pem" --channelID mychannel --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
peer lifecycle chaincode querycommitted --channelID mychannel --name basic --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/or
derer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
2025-03-30 15:53:50.856 CEST 0001 INFO [chaincodeCmd] ClientWait -> txid [68f69d031411fcd27623a323e2eed3874239020c268f6e0ae3f8b279c215f0c2] committed
with status (VALID) at localhost:11051
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true]
```

Ilustración 1. Org3MSP: False

Vemos que ya tenemos todos los Approvals de las tres organizaciones y comprobamos si ya podemos enviar transacciones.

4. Probamos la red

A continuación, vamos a enviar una transacción a modo de ejemplo con el siguiente comando:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" --peerAddresses localhost:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
```

Que debería devolver un 200, pero está dando el siguiente error:

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.
com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C m
ychannel -n basic --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.
com/tls/ca.crt" --peerAddresses localhost:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.c
om/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
Error: endorsement failure during invoke. response: status:500 message:"make sure the chaincode basic has been successfully defined on channel mychann
el and try again: chaincode definition for 'basic' exists, but chaincode is not installed"
```

Repetimos el proceso, a ver si conseguimos solventarlo.

Probamos con el siguente comando:

```
peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name basic --version 1.0 --package-id=6a6f743366675481339c7ddda5f281a441c9ad8c13e32a9a5c07b892e44de105 --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

Ahora parece, que ya está bien definido:

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOver
de orderer.example.com --channelID mychannel --name basic --version 1.0 --package-id=6a6f743366675481339c7ddda5f281a441c9ad8c13e32a9a5c07b892e44de10
--sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert
em
2025-03-30 16:19:08.797 CEST 0001 INFO [chaincodeCmd] ClientWait -> txid [0b1dddc906e2f21290a87d19ca8dbf0e170885fecadd2e9f05ee0f3573a02b22] committe
with status (VALID) at localhost:7051
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer lifecycle chaincode querycommitted -C mychannel --name basic
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true]
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$
```

Pero, si procedemos a ejecutar la transacción sigue fallando

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.
com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C myc
hannel -n basic -c '{"Args":["InitLedger"]}'
Error: endorsement failure during invoke. response: status:500 message:"make sure the chaincode basic has been successfully defined on channel mychann
el and try again: chaincode definition for 'basic' exists, but chaincode is not installed"
```

Volvemos a repetir el proceso:

```
peer lifecycle chaincode approveformyorg -o localhost:7050 --
ordererTLSHostnameOverride orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.exampl
e.com/msp/tlscacerts/tlsca.example.com-cert.pem" --channelID mychannel --name
basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
peer lifecycle chaincode querycommitted --channelID mychannel --name basic --
cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.exampl
e.com/msp/tlscacerts/tlsca.example.com-cert.pem"
```

Y ahora sí verificamos que la transacción se ha realizado con éxito:

```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.exampl
e.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --
peerAddresses localhost:9051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.exa
mple.com/tls/ca.crt" --peerAddresses localhost:11051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.exa
mple.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
```

```
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.
com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C m
ychannel -n basic --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.
com/tls/ca.crt" --peerAddresses localhost:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.c
om/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
2025-03-30 16:31:49.692 CEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
ubuntu@ubuntu-VirtualBox:~/Projects/fabric-samples/test-network$
```

```
peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}' | jq
```

```
2025-03-30 16:33:12.029 CEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
[
  {
    "AppraisedValue": 300,
    "Color": "blue",
    "ID": "asset1",
    "Owner": "Tomoko",
    "Size": 5
  },
  {
    "AppraisedValue": 400,
    "Color": "red",
    "ID": "asset2",
    "Owner": "Brad",
    "Size": 5
  },
  {
    "AppraisedValue": 500,
    "Color": "green",
    "ID": "asset3",
    "Owner": "Jin Soo",
    "Size": 10
  },
  {
    "AppraisedValue": 600,
    "Color": "yellow",
    "ID": "asset4",
    "Owner": "Max",
    "Size": 10
  },
  {
    "AppraisedValue": 700,
    "Color": "black",
    "ID": "asset5",
```

5. Conclusiones

A lo largo de esta práctica, he podido familiarizarme con la estructura y funcionamiento de Hyperledger Fabric, logrando tres objetivos principales:

1. **Incorporar una nueva organización a la red**
2. **Desplegar un nuevo chaincode**
3. **Ejecutar transacciones en la nueva test-network modificada**

Por el contrario, no se han podido llegar a completar otros aspectos como la modificación o incorporación de nuevos chaincodes con la lógica de negocio requerido para el caso de uso planteado o añadir nuevos peers a las organizaciones.

No obstante, en el módulo se ha comprendido que Hyperledger Fabric se basa en una arquitectura modular donde cada organización opera de manera independiente dentro de la red.

La incorporación de una nueva organización (**Distribuidor**) requirió la modificación de múltiples archivos de configuración (configtx.yaml, docker-compose.yaml, etc.), la generación de certificados de identidad y la actualización del canal para incluirla correctamente.

Este proceso refleja **la flexibilidad de Fabric**, pero también la **necesidad** y **complejidad** de una **configuración** precisa para garantizar la interoperabilidad de los distintos participantes.

La práctica me ha permitido comprender los principios fundamentales de Hyperledger Fabric, desde su configuración inicial hasta la ejecución de transacciones. La complejidad del proceso resalta la importancia de la planificación de la configuración a realizar y el seguimiento meticuloso de los procedimientos para garantizar el correcto funcionamiento de la red. Ahora que he logrado desplegar una red funcional con

múltiples organizaciones y transacciones operativas, el siguiente paso sería explorar aspectos avanzados como añadir nuevos peers a las organizaciones, visualizar las transacciones por un explorador de bloques de fabric o ejecutar las transacciones mediante API vía POSTMAN entre otras cosas.