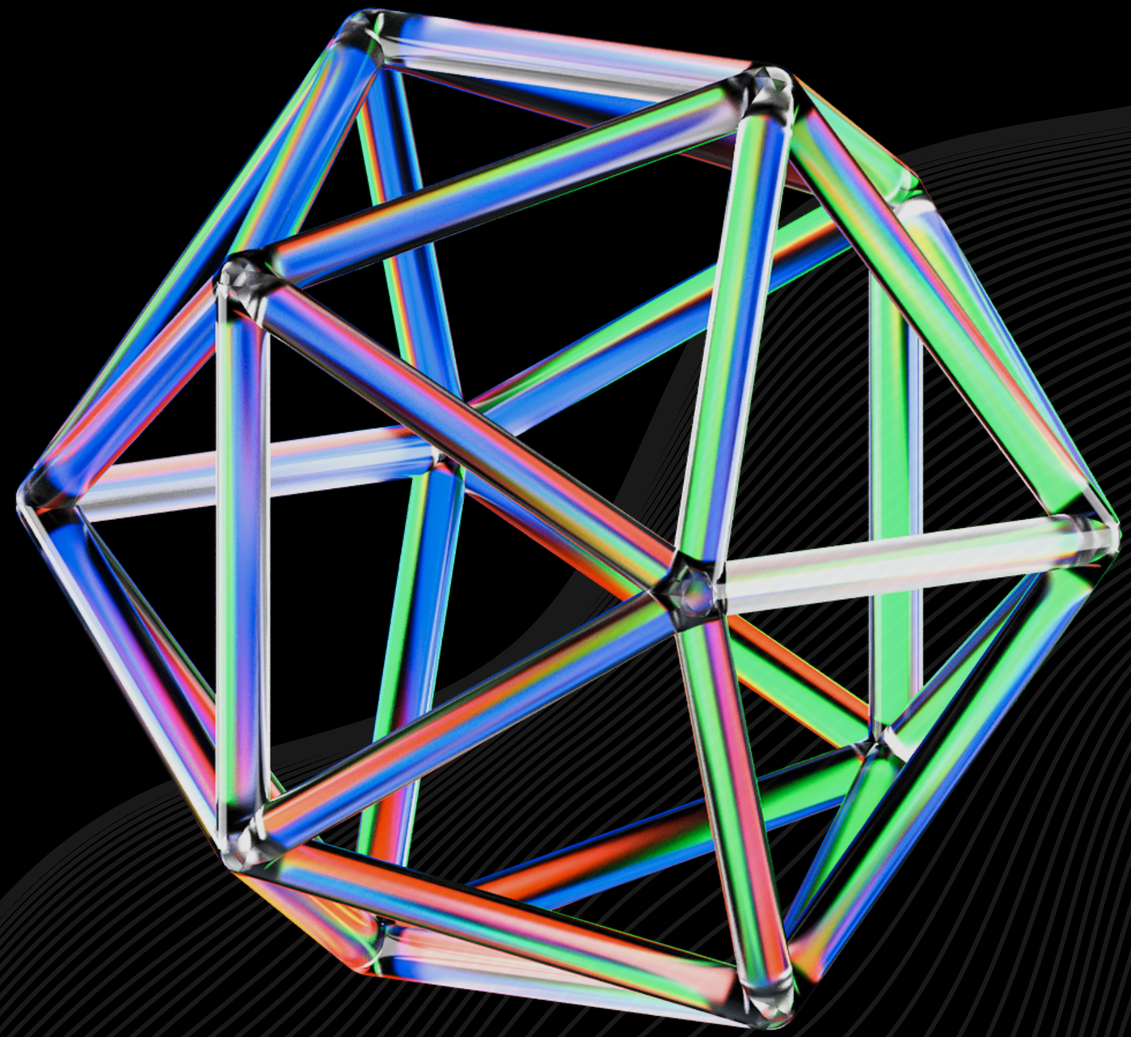
PROYECTO FINAL

HYPERLEDGER FABRIC

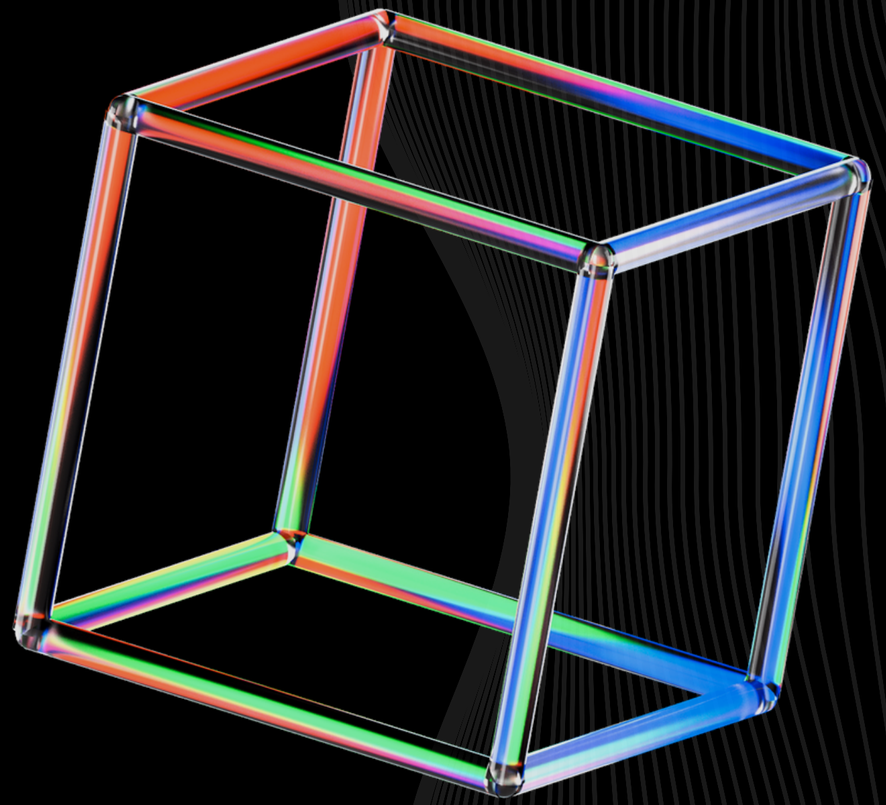


POR: KEISY MORALES,
JULIO MOU,
RODOLFO VELAZQUEZ

2022

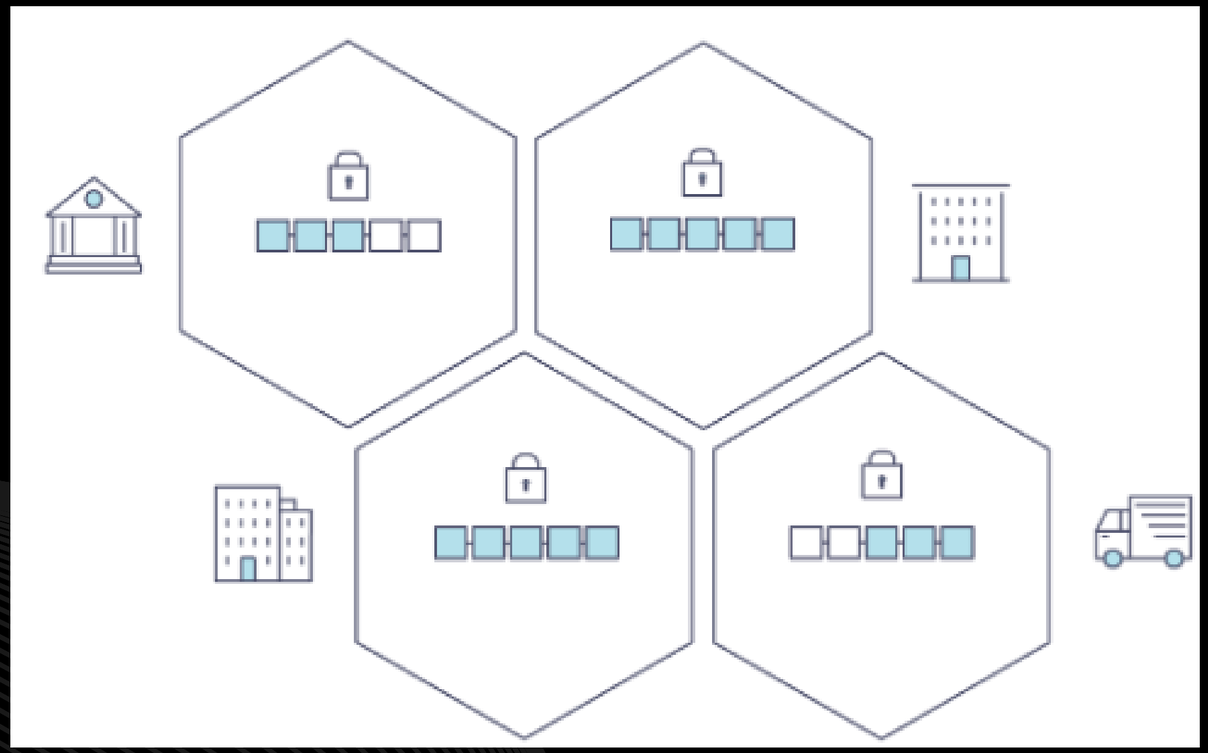
INTRODUCCIÓN

En términos generales, una blockchain (cadena de bloques) es un libro de transacciones inmutable, mantenido dentro de una red distribuida de nodos pares. Cada uno de estos nodos mantiene una copia del libro mayor aplicando transacciones que han sido validadas por un protocolo de consenso, agrupadas en bloques que incluyen un hash que vincula cada bloque al bloque precedente.



¿Qué es una Blockchain?

- UN LIBRO MAYOR DISTRIBUIDO



Un libro de contabilidad de blockchain suele describirse como descentralizado porque se replica en muchos participantes de la red, cada uno de los cuales colabora en su mantenimiento. Veremos que la descentralización y la colaboración son atributos poderosos que reflejan la forma en que las empresas intercambian bienes y servicios en el mundo real.

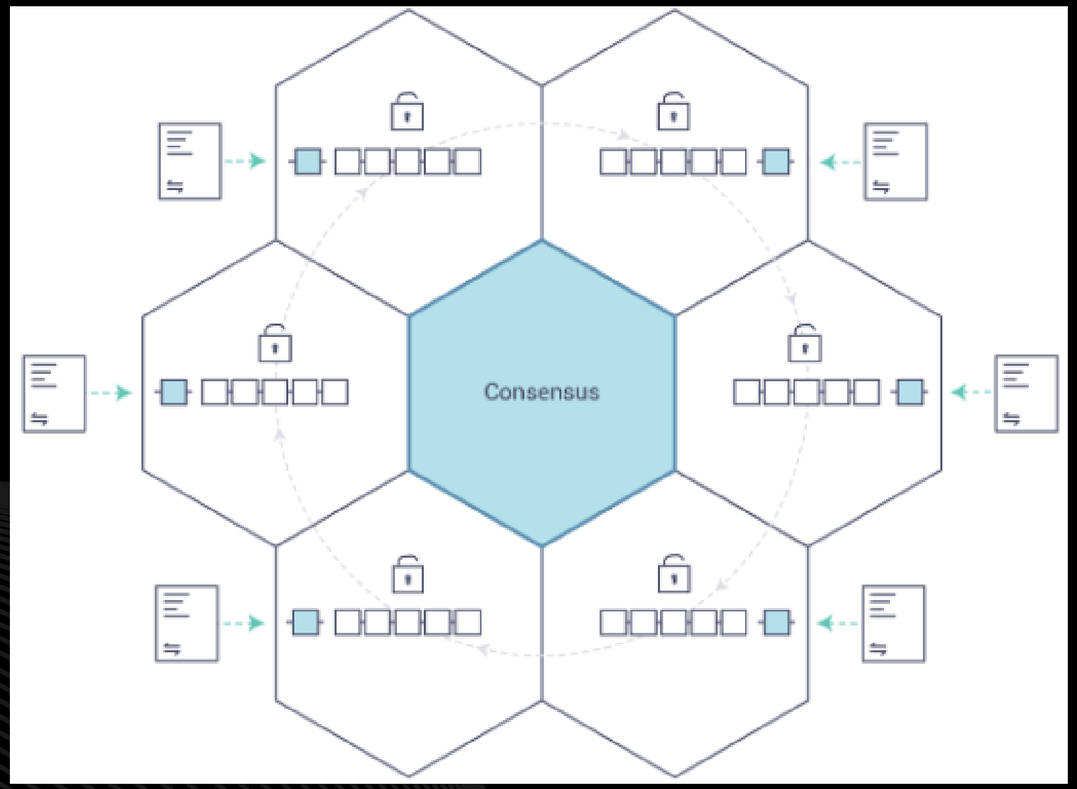
¿Qué es una Blockchain?

- CONTRATOS INTELIGENTES

Para apoyar la actualización consistente de la información - y para permitir toda una serie de funciones del libro mayor (transacciones, consultas, etc.) - una red blockchain utiliza contratos inteligentes para proporcionar un acceso controlado al libro mayor.

¿Qué es una Blockchain?

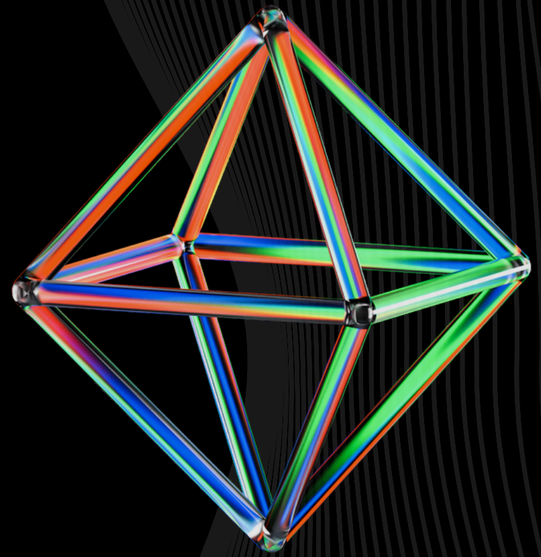
- CONSENSO



El proceso de mantener las transacciones del libro mayor sincronizadas en toda la red -para garantizar que los libros mayores se actualizan sólo cuando las transacciones son aprobadas por los participantes apropiados, y que cuando los libros mayores se actualizan, lo hacen con las mismas transacciones en el mismo orden- se llama consenso.

¿Qué es Hyperledger Fabric?

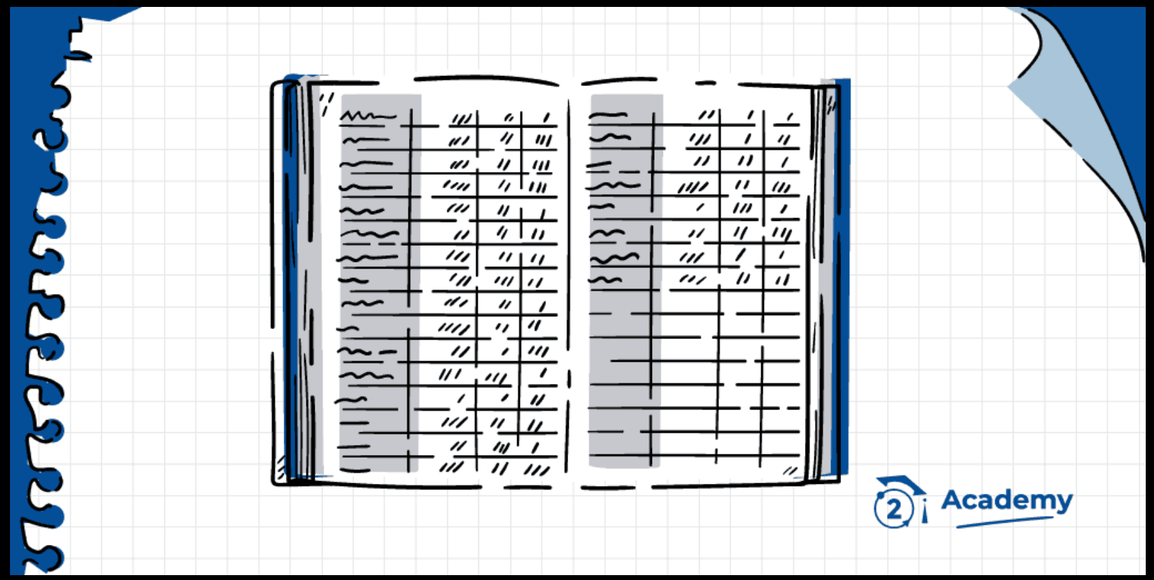
Hyperledger Fabric es una plataforma para soluciones de libro mayor distribuido basada en una arquitectura modular que ofrece altos grados de confidencialidad, resistencia, flexibilidad y escalabilidad. Está diseñada para soportar implementaciones enchufables de diferentes componentes y acomodar la complejidad y las complejidades que existen en todo el ecosistema económico.

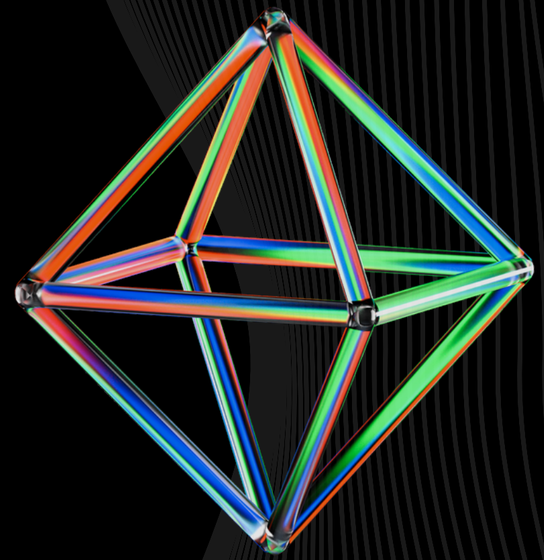


HYPERLEDGER
FABRIC

Libro Mayor Compartido

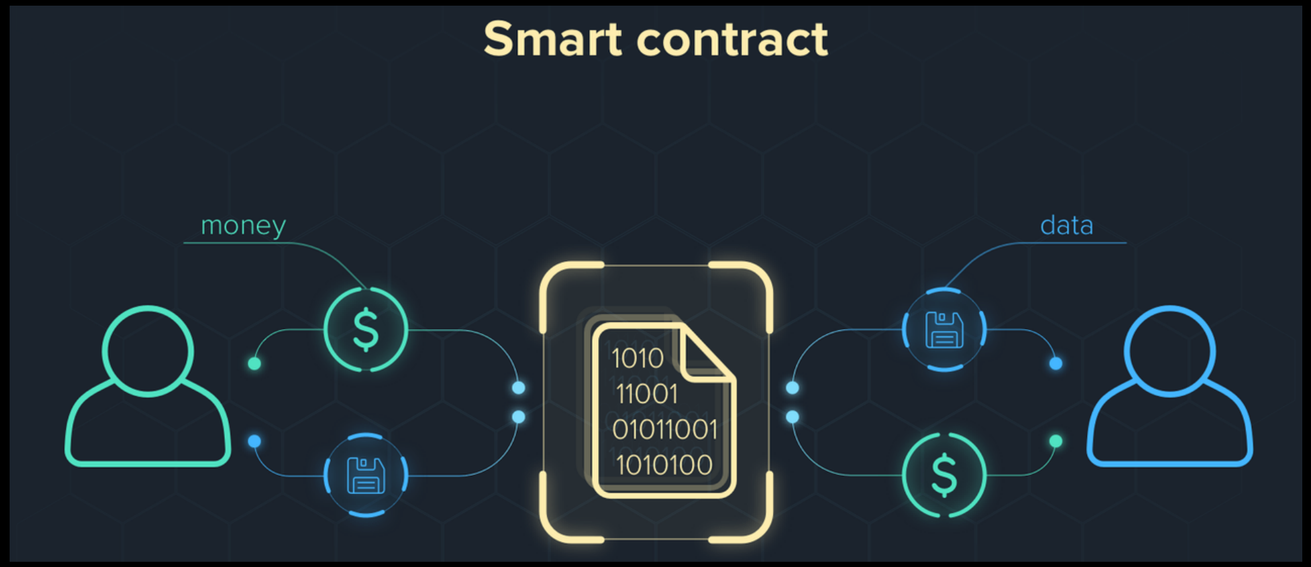
Hyperledger Fabric tiene un subsistema de libro mayor que comprende dos componentes: el estado mundial y el registro de transacciones. Cada participante tiene una copia del libro mayor de cada red Hyperledger Fabric a la que pertenece.

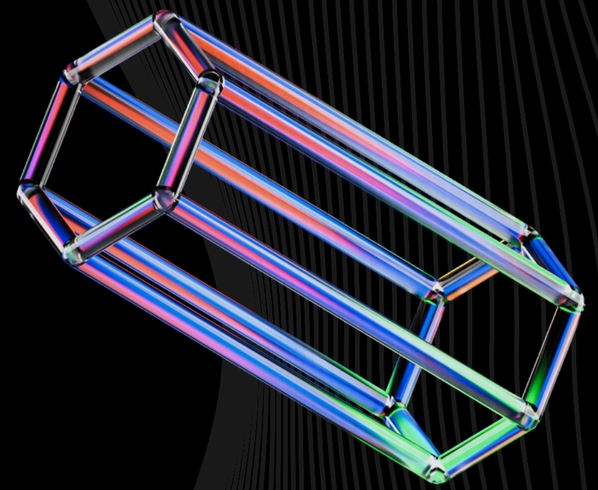




Contratos inteligentes

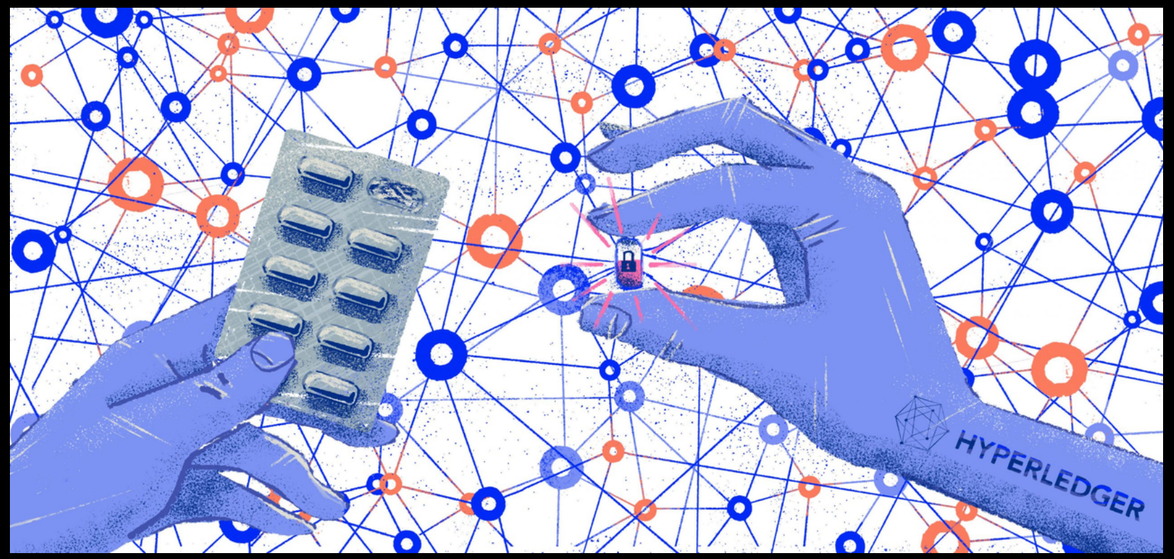
Los contratos inteligentes de Hyperledger Fabric se escriben en chaincode y son invocados por una aplicación externa a la cadena de bloques cuando esa aplicación necesita interactuar con el libro mayor.





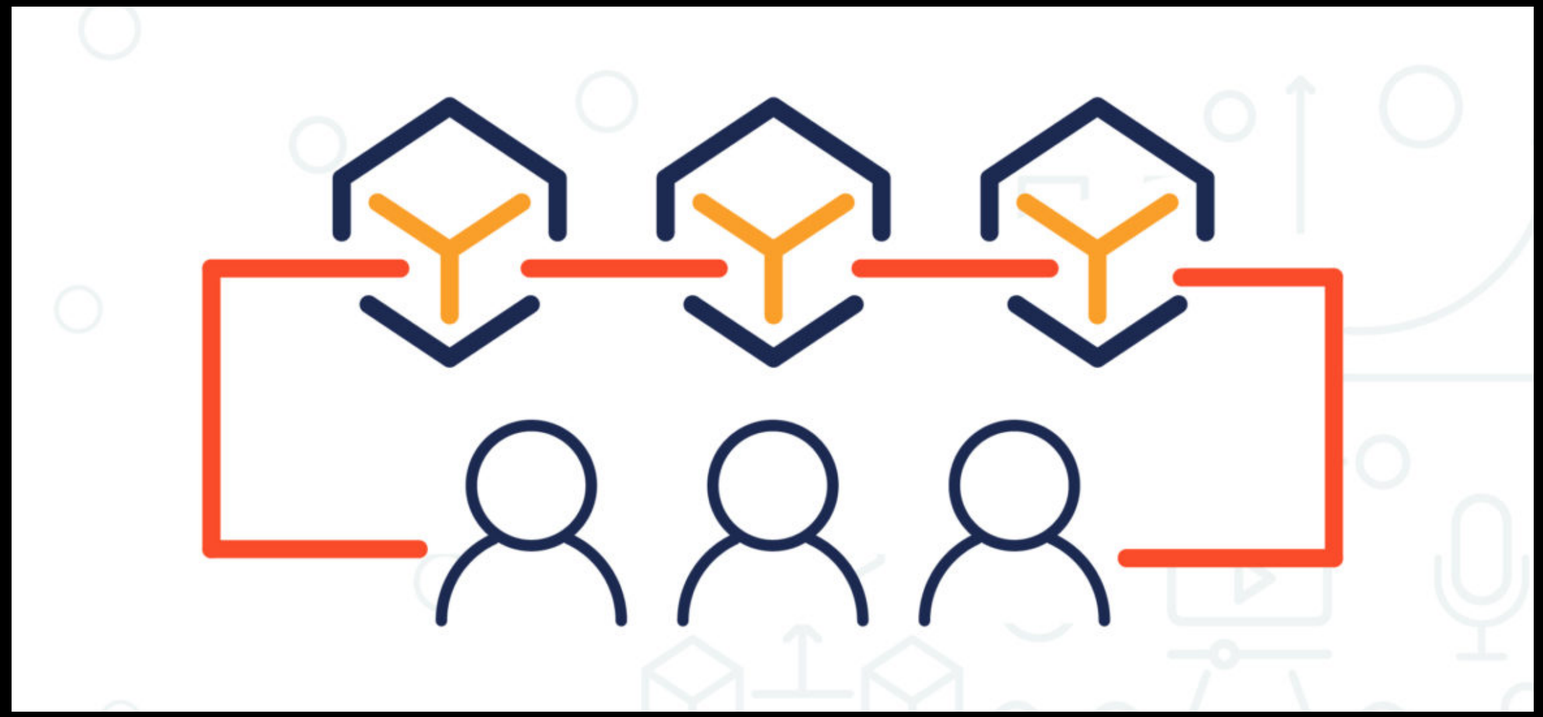
Privacidad

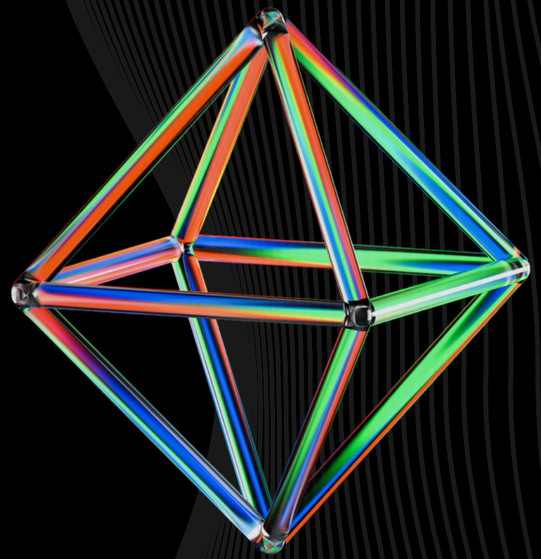
Hyperledger Fabric es compatible con redes en las que la privacidad (mediante canales) es un requisito operativo clave, así como con redes que son comparativamente abiertas.



Consenso

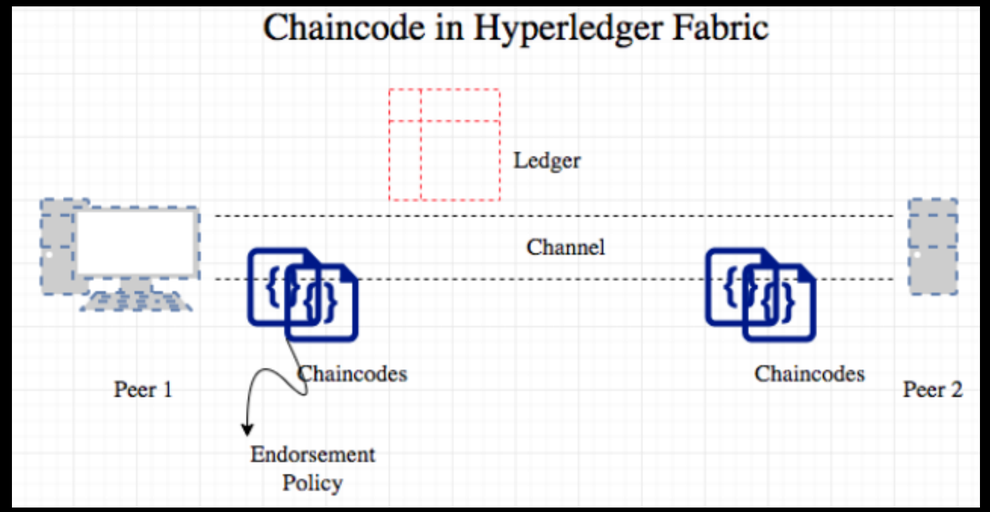
Hyperledger Fabric ha sido diseñado para permitir a los iniciadores de la red elegir el mecanismo de consenso que mejor represente las relaciones que existen entre los participantes.





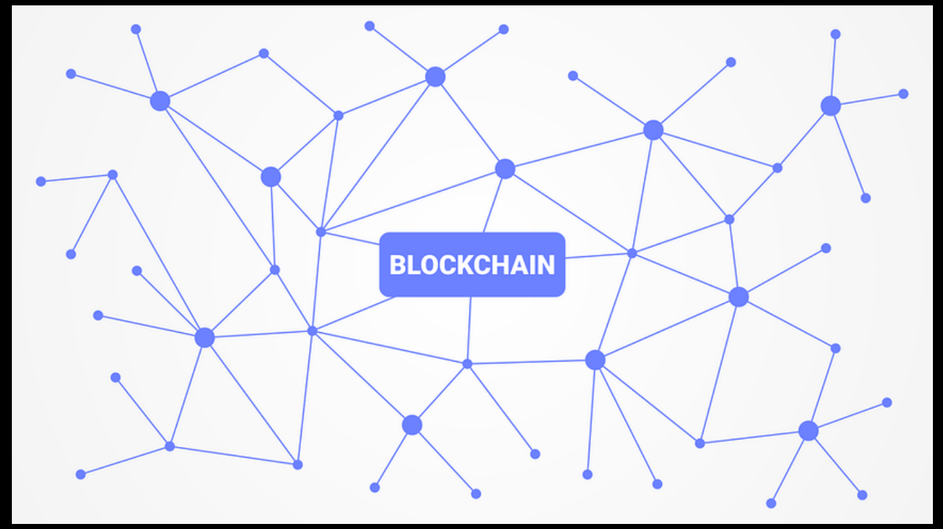
Chaincode

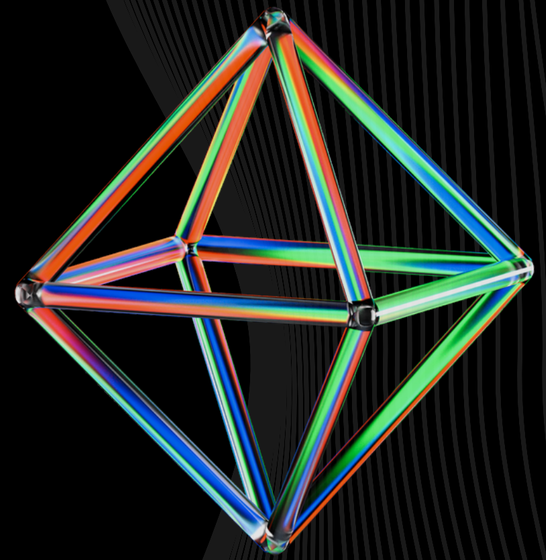
El Chaincode hace cumplir las reglas para leer o modificar los pares clave-valor u otra información de la base de datos de estado. Las funciones de Chaincode se ejecutan contra la base de datos del estado actual del libro mayor y se inician mediante una propuesta de transacción. La ejecución del Chaincode da como resultado un conjunto de escrituras clave-valor (conjunto de escritura) que puede ser enviado a la red y aplicado al libro mayor en todos los pares.



¿Qué es una blockchain network?

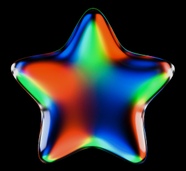
Una red blockchain es una infraestructura técnica que proporciona servicios de libro mayor y contratos inteligentes (chaincode) a las aplicaciones. Principalmente, los contratos inteligentes se utilizan para generar transacciones que posteriormente se distribuyen a todos los nodos pares de la red, donde se registran de forma inmutable en su copia del libro mayor.



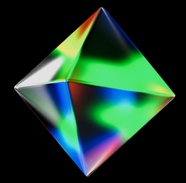


Getting Started- Instalar

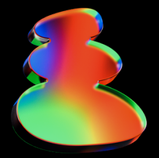
La pila de aplicaciones de Fabric tiene cinco capas:



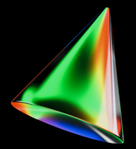
- **Software de requisito previo**: la capa base necesaria para ejecutar el software, por ejemplo, Docker.

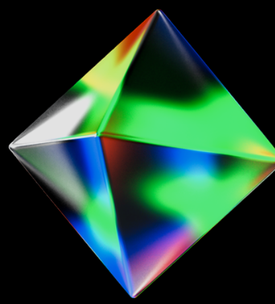


- **Muestras de Fabric y Fabric :** los ejecutables de Fabric para ejecutar una red Fabric junto con código de muestra.
- **Contract APIs :** para desarrollar contratos inteligentes ejecutados en una Fabric Network.

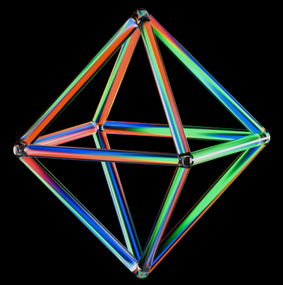


- **API de aplicaciones :** para desarrollar su aplicación blockchain.
- **La aplicación:** su aplicación de cadena de bloques utilizará los SDK de la aplicación para llamar a contratos inteligentes que se ejecutan en una red Fabric.



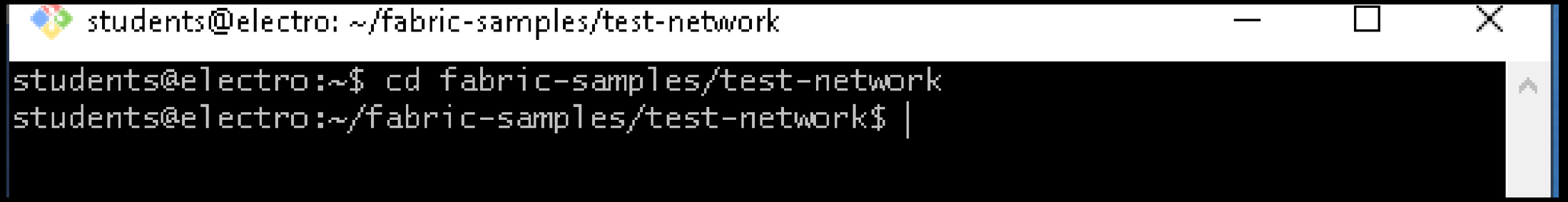


Using the Fabric test network

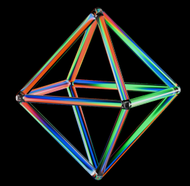


- **Abrir la red de prueba**

cd fabric-samples/test-network



Using the Fabric test network

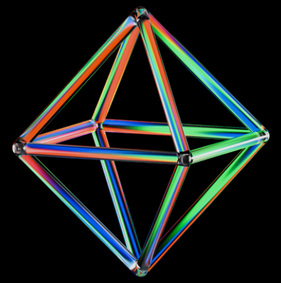


- **Abrir la red de prueba**

Se puede ejecutar para imprimir el texto de ayuda del script: `./network.sh -h`

```
students@electro: ~/fabric-samples/test-network
students@electro:~$ cd fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ ./network.sh -h
Using docker and docker-compose
Usage:
  network.sh <Mode> [Flags]
    Modes:
      up - Bring up Fabric orderer and peer nodes. No channel is created
      up createChannel - Bring up fabric network with one channel
      createChannel - Create and join a channel after the network is created
      deployCC - Deploy a chaincode to a channel (defaults to asset-transfer-bas
ic)
      down - Bring down the network

    Flags:
    Used with network.sh up, network.sh createChannel:
    -ca <use CAs> -  Use Certificate Authorities to generate network crypto mate
rial
    -c <channel name> - Name of channel to create (defaults to "mychannel")
    -s <dbtype> - Peer state database to deploy: goleveldb (default) or couchdb
    -r <max retry> - CLI times out after certain number of attempts (defaults to
 5)
    -d <delay> - CLI delays for a certain number of seconds (defaults to 3)
    -verbose - Verbose mode

    Used with network.sh deployCC
    -c <channel name> - Name of channel to deploy chaincode to
    -ccn <name> - Chaincode name.
    -ccl <language> - Programming language of the chaincode to deploy: go, java,
 javascript, typescript
    -ccv <version>  - Chaincode version. 1.0 (default), v2, version3.x, etc
    -ccs <sequence>  - Chaincode definition sequence. Must be an integer, 1 (def
ault), 2, 3, etc
    -ccp <path>  - File path to the chaincode.
    -ccep <policy>  - (Optional) Chaincode endorsement policy using signature po
licy syntax. The default policy requires an endorsement from Org1 and Org2
    -cccg <collection-config>  - (Optional) File path to private data collection
s configuration file
    -cci <fcn name>  - (Optional) Name of chaincode initialization function. Whe
n a function is provided, the execution of init will be requested and the functi
on will be invoked.

    -h - Print this message

 Possible Mode and flag combinations
   up -ca -r -d -s -verbose
   up createChannel -ca -c -r -d -s -verbose
   createChannel -c -r -d -verbose
   deployCC -ccn -ccl -ccv -ccs -ccp -cci -r -d -verbose
```

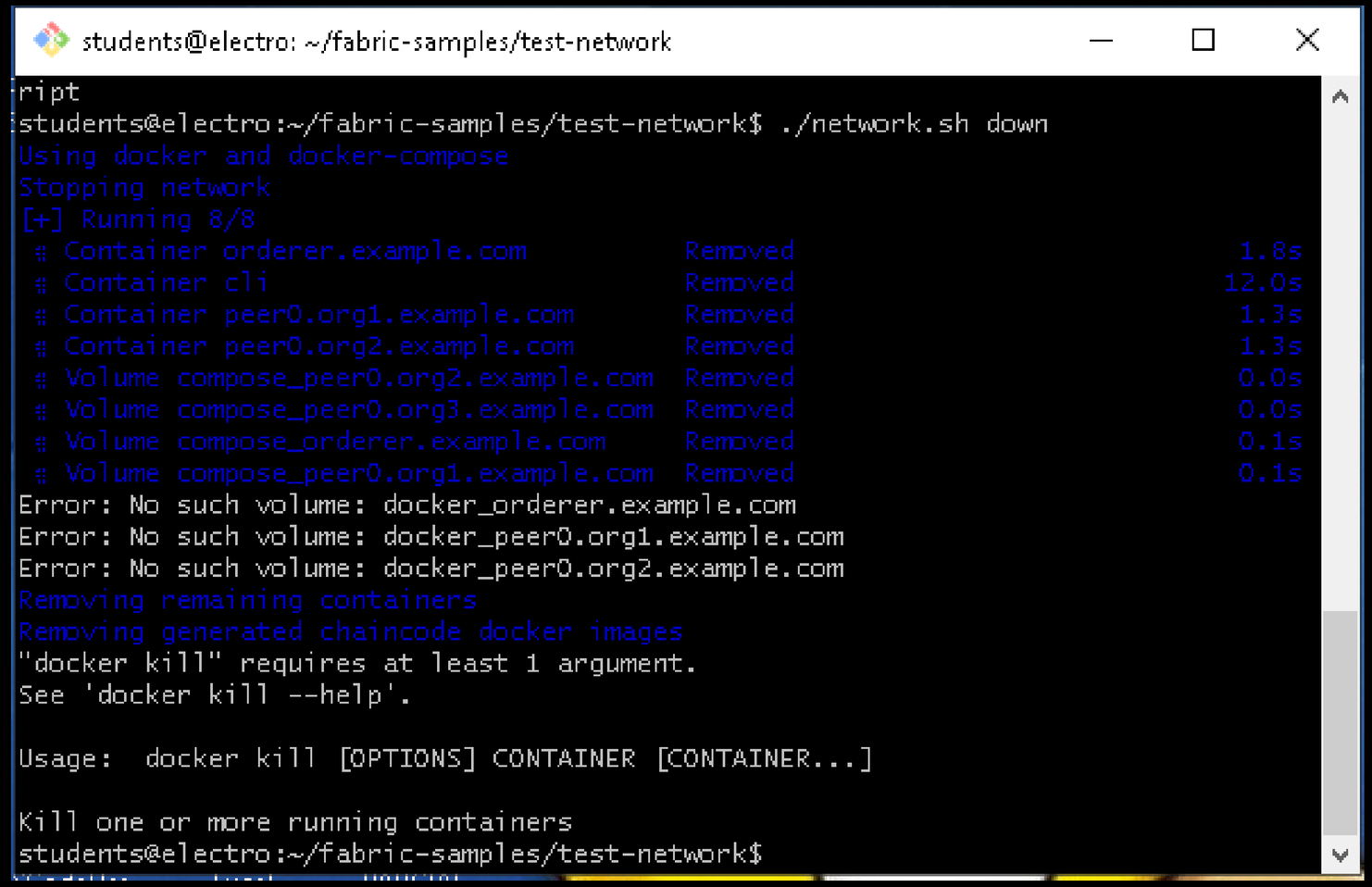
Using the Fabric test network



- **Abrir la red de prueba**

Desde dentro del test-network directorio, ejecute el siguiente comando para eliminar cualquier contenedor o artefacto de ejecuciones anteriores:

./network.sh down

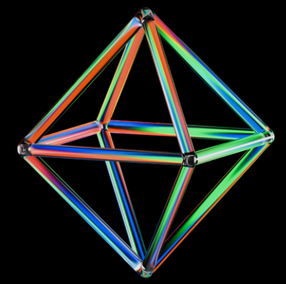


Using the Fabric test network

- **Abrir la red de prueba**

Luego se puede abrir la red emitiendo el siguiente comando.

./network.sh up



```
students@electro: ~/fabric-samples/test-network
students@electro:~/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 us
ing database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
/home/students/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.ya
ml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
[+] Running 7/7
 ⠿ Volume "compose_orderer.example.com"     Created                          0.0s
 ⠿ Volume "compose_peer0.org1.example.com"  Created                          0.0s
 ⠿ Volume "compose_peer0.org2.example.com"  Created                          0.1s
 ⠿ Container peer0.org2.example.com         Started                          5.6s
 ⠿ Container orderer.example.com            Started                          4.3s
 ⠿ Container peer0.org1.example.com         Started                          5.6s
 ⠿ Container cli                            Started                          5.1s
CONTAINER ID   IMAGE                               COMMAND             CREATED
       STATUS                  PORTS

 NAMES
519667d9ae70   hyperledger/fabric-tools:latest     "/bin/bash"         5 seconds
 ago   Up Less than a second

 cli
03c9d65ec696   hyperledger/fabric-peer:latest      "peer node start"   7 seconds
 ago   Up 1 second             0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/t
cp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp
 peer0.org2.example.com
8f9ad6249586   hyperledger/fabric-orderer:latest   "orderer"           7 seconds
 ago   Up 3 seconds            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
203bb26d3ded   hyperledger/fabric-peer:latest      "peer node start"   7 seconds
 ago   Up 1 second             0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.
0:9444->9444/tcp, :::9444->9444/tcp
```

Using the Fabric test network

- Los componentes de la red de prueba.

Ejecutar el siguiente comando para enumerar todos los contenedores de Docker que se ejecutan en su máquina:

docker ps -a

```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ docker ps -a
CONTAINER ID   IMAGE                               COMMAND                CREATED
        STATUS         PORTS
                                                                              NAMES
519667d9ae70   hyperledger/fabric-tools:latest     "/bin/bash"            5 minute
s ago     Up 5 minutes
                                                                              cli
03c9d65ec696   hyperledger/fabric-peer:latest      "peer node start"      5 minute
s ago     Up 5 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
8f9ad6249586   hyperledger/fabric-orderer:latest   "orderer"              5 minute
s ago     Up 5 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   order
er.example.com
203bb26d3ded   hyperledger/fabric-peer:latest      "peer node start"      5 minute
s ago     Up 5 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
students@electro:~/fabric-samples/test-network$
```

Cada nodo y usuario que interactúa con una red Fabric debe pertenecer a una organización que sea miembro de la red. El grupo de organizaciones que son miembros de una red Fabric a menudo se denomina consorcio. La red de prueba tiene dos miembros del consorcio, Org1 y Org2. La red también incluye una organización de pedidos que mantiene el servicio de pedidos de la red.

Using the Fabric test network

- Creando un canal

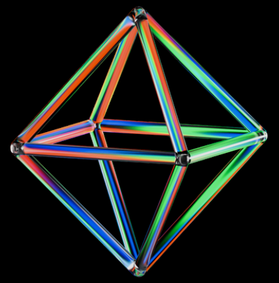
Se ejecute el siguiente comando para crear un canal con el nombre predeterminado de **mychannel**:

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

Si el comando fue exitoso, puede ver el siguiente mensaje impreso en sus registros:

```
2022-08-25 03:07:11.834 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-08-25 03:07:11.982 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
```

Using the Fabric test network



- Creando un canal

Como ejemplo, el siguiente comando crearía un canal llamado **channel1:**

```
./network.sh createChannel -c channel1
```

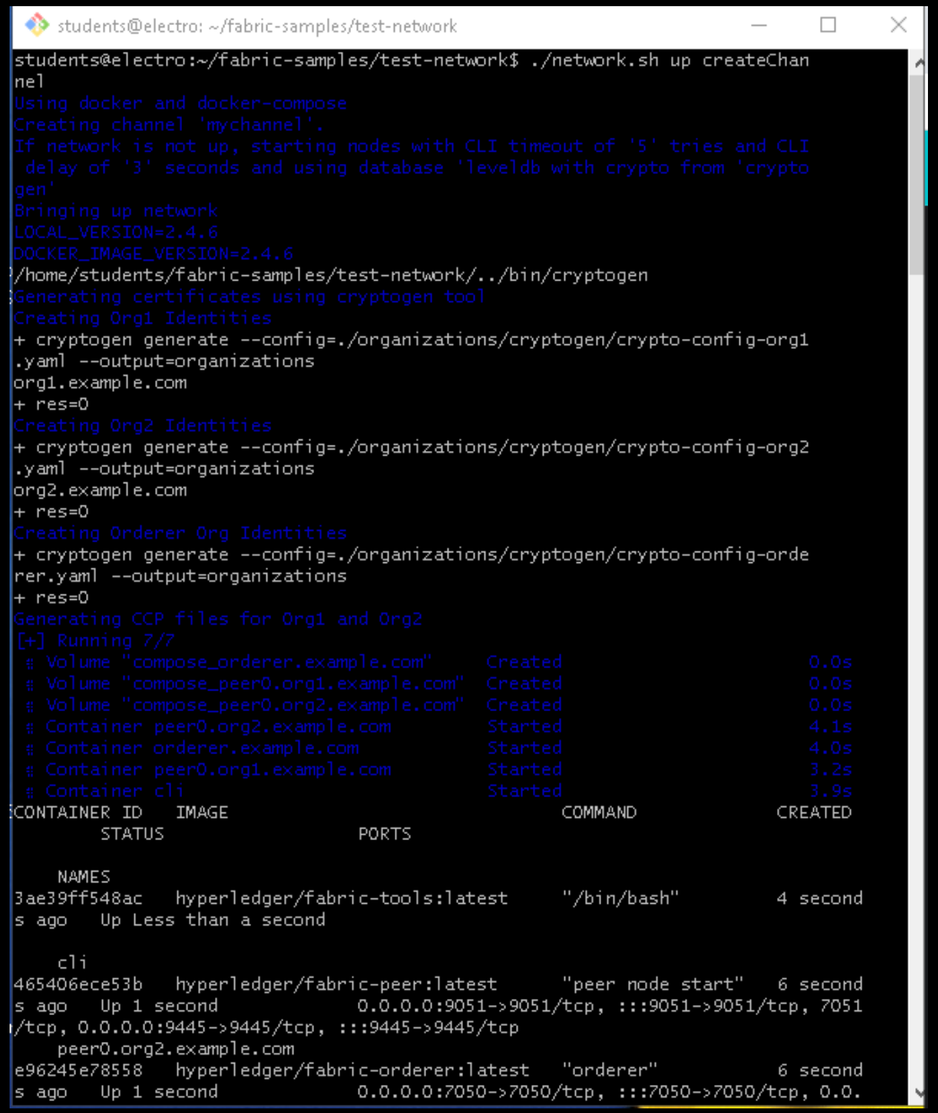
Si el comando fue exitoso, puede ver el siguiente mensaje impreso en sus registros:

```
2022-08-25 03:13:35.887 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and o
rderer connections initialized
2022-08-25 03:13:35.905 UTC 0002 INFO [channelCmd] update -> Successfully submitted
 channel update
Anchor peer set for org 'Org2MSP' on channel 'channel1'
Channel 'channel1' joined
```

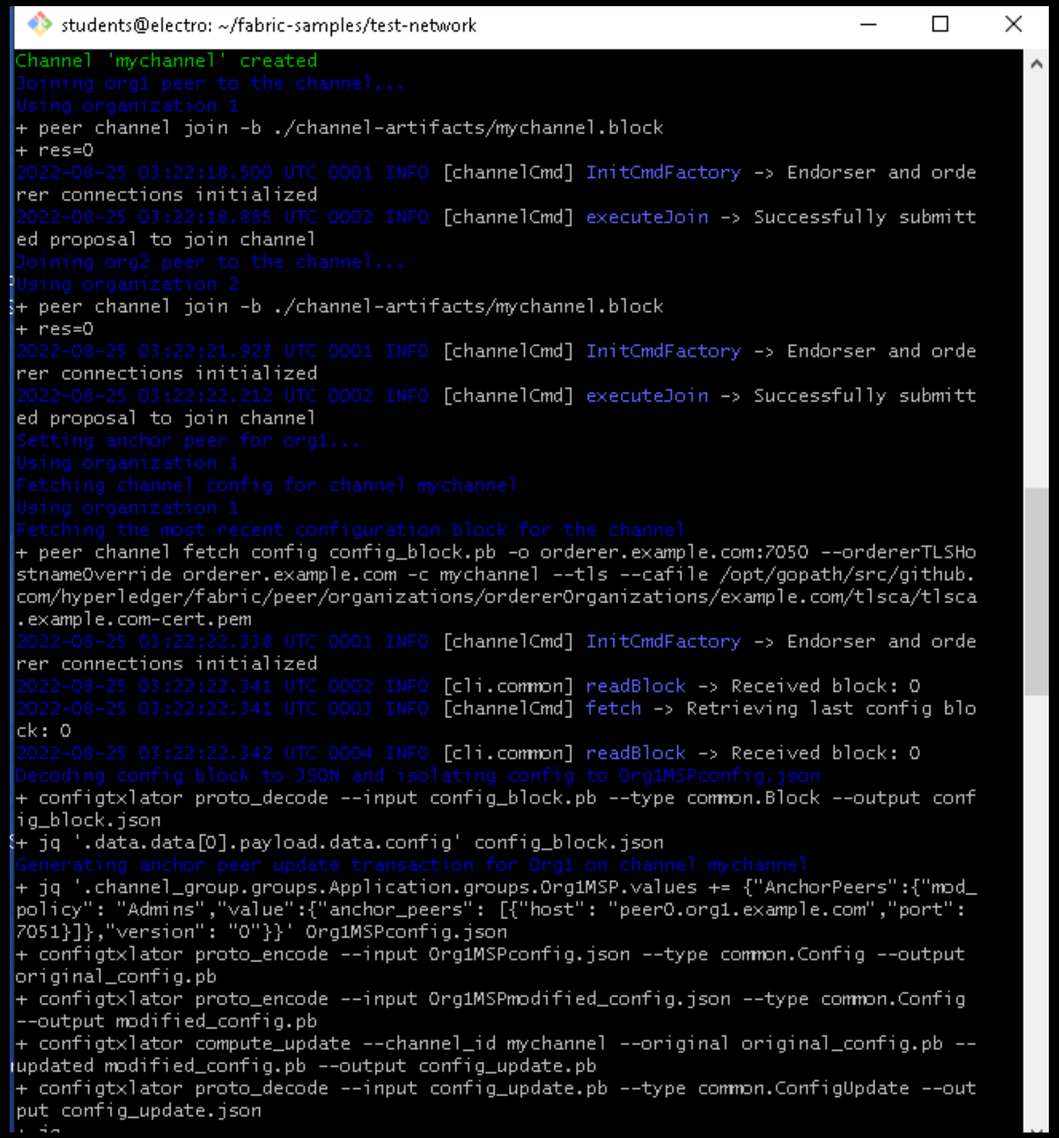
- Creando un canal

Para abrir la red y crear un canal en un solo paso, puede usar los modos **up** y juntos: **createChannel**

./network.sh up createChannel



```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ ./network.sh up createChan
nel
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI
 delay of '3' seconds and using database 'leveldb with crypto from 'crypto
gen'
Bringing up network
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
/home/students/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-orde
rer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
[+] Running 7/7
 ⠿ Volume "compose_orderer.example.com"     Created                    0.0s
 ⠿ Volume "compose_peer0.org1.example.com"  Created                    0.0s
 ⠿ Volume "compose_peer0.org2.example.com"  Created                    0.0s
 ⠿ Container peer0.org2.example.com         Started                    4.1s
 ⠿ Container orderer.example.com            Started                    4.0s
 ⠿ Container peer0.org1.example.com         Started                    3.2s
 ⠿ Container cli                            Started                    3.9s
CONTAINER ID   IMAGE                               COMMAND             CREATED
       STATUS                  PORTS

   NAMES
3ae39ff548ac   hyperledger/fabric-tools:latest     "/bin/bash"         4 second
s ago   Up Less than a second

   cli
465406ece53b   hyperledger/fabric-peer:latest      "peer node start"   6 second
s ago   Up 1 second             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
e96245e78558   hyperledger/fabric-orderer:latest   "orderer"           6 second
s ago   Up 1 second             0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.
```



```
students@electro: ~/fabric-samples/test-network
Channel 'mychannel' created
Joining org1 peer to the channel...
Using organization 1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2022-08-25 03:22:18.500 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orde
rer connections initialized
2022-08-25 03:22:18.885 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitt
ed proposal to join channel
Joining org2 peer to the channel...
Using organization 2
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2022-08-25 03:22:21.923 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orde
rer connections initialized
2022-08-25 03:22:22.212 UTC 0002 INFO [channelCmd] executeJoin -> Successfully submitt
ed proposal to join channel
Setting anchor peer for org1...
Using organization 1
Fetching channel config for channel mychannel
Using organization 1
Fetching the most recent configuration block for the channel
+ peer channel fetch config config_block.pb -o orderer.example.com:7050 --ordererTLSHo
stnameOverride orderer.example.com -c mychannel --tls --cafile /opt/gopath/src/github.
com/hyperledger/fabric/peer/organizations/ordererOrganizations/example.com/tlsca/tlsca
.example.com-cert.pem
2022-08-25 03:22:22.338 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orde
rer connections initialized
2022-08-25 03:22:22.341 UTC 0002 INFO [cli.common] readBlock -> Received block: 0
2022-08-25 03:22:22.341 UTC 0003 INFO [channelCmd] fetch -> Retrieving last config blo
ck: 0
2022-08-25 03:22:22.342 UTC 0004 INFO [cli.common] readBlock -> Received block: 0
Decoding config block to JSON and isolating config to Org1MSPconfig.json
+ configtxlator proto_decode --input config_block.pb --type common.Block --output conf
ig_block.json
+ jq '.data.data[0].payload.data.config' config_block.json
Generating anchor peer update transaction for Org1 on channel mychannel
+ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_
policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1.example.com","port":
7051}]},"version": "0"}}' Org1MSPconfig.json
+ configtxlator proto_encode --input Org1MSPconfig.json --type common.Config --output
original_config.pb
+ configtxlator proto_encode --input Org1MSPmodified_config.json --type common.Config
--output modified_config.pb
+ configtxlator compute_update --channel_id mychannel --original original_config.pb --
updated modified_config.pb --output config_update.pb
+ configtxlator proto_decode --input config_update.pb --type common.ConfigUpdate --out
put config_update.json
```

- Creando un canal

Para abrir la red y crear un canal en un solo paso, puede usar los modos **up** y juntos: **createChannel**

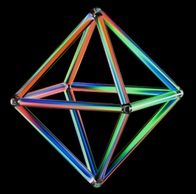
```
students@electro: ~/fabric-samples/test-network
67acbd6e0279   hyperledger/fabric-peer:latest      "peer node start"   6 second
s ago   Up 2 seconds           0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.
0.0:9444->9444/tcp, :::9444->9444/tcp
    peer0.org1.example.com
Using docker and docker-compose
Generating channel genesis block 'mychannel.block'
/home/students/fabric-samples/test-network/../bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifac
ts/mychannel.block -channelID mychannel
2022-08-25 03:22:12.148 UTC 0001 INFO [common.tools.configtxgen] main -> Loadin
g configuration
2022-08-25 03:22:12.154 UTC 0002 INFO [common.tools.configtxgen.localconfig] co
mpleteInitialization -> orderer type: etcdraft
2022-08-25 03:22:12.154 UTC 0003 INFO [common.tools.configtxgen.localconfig] co
mpleteInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interva
l:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_inte
rval_size:16777216
2022-08-25 03:22:12.154 UTC 0004 INFO [common.tools.configtxgen.localconfig] Lo
ad -> Loaded configuration: /home/students/fabric-samples/test-network/configtx
/configtx.yaml
2022-08-25 03:22:12.156 UTC 0005 INFO [common.tools.configtxgen] doOutputBlock
-> Generating genesis block
2022-08-25 03:22:12.156 UTC 0006 INFO [common.tools.configtxgen] doOutputBlock
-> Creating application channel genesis block
2022-08-25 03:22:12.156 UTC 0007 INFO [common.tools.configtxgen] doOutputBlock
-> Writing genesis block
+ res=0
Creating channel mychannel
Using organization 1
+ osnadmin channel join --channelID mychannel --config-block ./channel-artifacts/my
nnel.block -o localhost:7053 --ca-file /home/students/fabric-samples/test-network/o
nizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --clien
ert /home/students/fabric-samples/test-network/organizations/ordererOrganizations/e
ple.com/orderers/orderer.example.com/tls/server.crt --client-key /home/students/fab
-samples/test-network/organizations/ordererOrganizations/example.com/orderers/order
example.com/tls/server.key
+ res=0
Status: 201
{
        "name": "mychannel",
        "url": "/participation/v1/channels/mychannel",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}
```

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

Si el comando fue exitoso, puede ver el siguiente mensaje impreso en sus registros:

```
2022-08-25 03:22:22.878 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser an
d orderer connections initialized
2022-08-25 03:22:22.898 UTC 0002 INFO [channelCmd] update -> Successfully submit
ted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
```

Using the Fabric test network



- Iniciar un chaincode en el canal

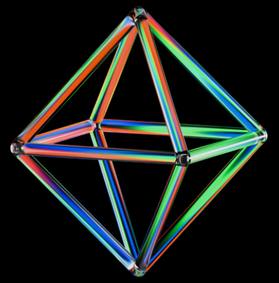
Después de haber usado **network.sh** para crear un canal, puede iniciar un chaincode en el canal usando el siguiente comando:

```
students@electro:~/fabric-samples/test-network$ ./network.sh deployCC -ccn basic
 -ccp ../asset-transfer-basic/chaincode-go -ccl go
Using docker and docker-compose
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-go
- CC_SRC_LANGUAGE: go
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
Vendoring Go dependencies at ../asset-transfer-basic/chaincode-go
~/fabric-samples/asset-transfer-basic/chaincode-go ~/fabric-samples/test-network
go: unknown subcommand "mod"
Run 'go help' for usage.
~/fabric-samples/test-network
Finished vendoring Go dependencies
+ peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/c
haincode-go --lang golang --label basic_1.0
+ res=1
Error: error getting chaincode bytes: listing deps for package ../asset-transfer
-basic/chaincode-go failed: exit status 2
Chaincode packaging has failed
Deploying chaincode failed
```

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

El **deployCC** subcomando instalará el código de cadena fabcar y **peer0.org1.example.com** luego **peer0.org2.example.com** implementará el código de cadena en el canal especificado usando el indicador de canal (o **mychannel** si no se especifica ningún canal).

Using the Fabric test network



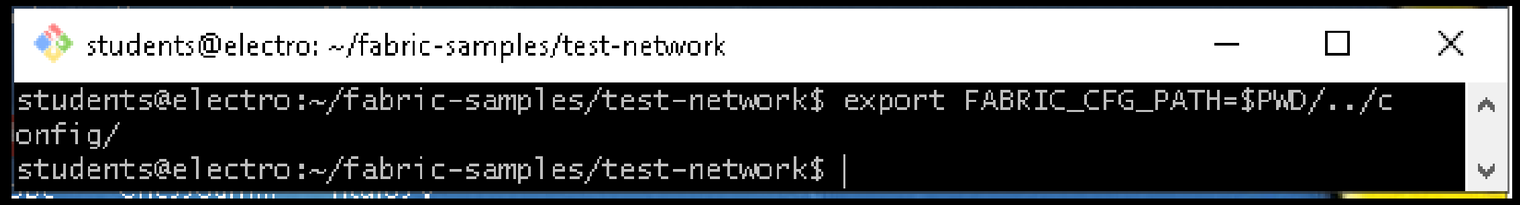
- Interactuando con la red

Se usa el siguiente comando para agregar esos archivos binarios a su ruta CLI:

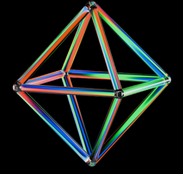
`export PATH=${PWD}/../bin:$PATH`

También debe configurar `FABRIC_CFG_PATH` para que apunte al `core.yaml` archivo en el `fabric-samples` repositorio:

`export FABRIC_CFG_PATH=$PWD/../config/`



Using the Fabric test network



- Interactuando con la red

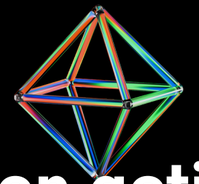
Ahora puede configurar las variables de entorno que le permiten operar la `peer` CLI como Org1:

```
# Environment variables for Org1
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

```
students@electro: ~/fabric-samples/test-network
onfig/
students@electro:~/fabric-samples/test-network$ export CORE_PEER_TLS_ENABLED=tru
e
students@electro:~/fabric-samples/test-network$ export CORE_PEER_LOCALMSPID="Org
1MSP"
students@electro:~/fabric-samples/test-network$ export CORE_PEER_TLS_ROOTCERT_FI
LE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.exam
ple.com/tls/ca.crt
students@electro:~/fabric-samples/test-network$ export CORE_PEER_MSPCONFIGPATH=$
{PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.
com/msp
students@electro:~/fabric-samples/test-network$ export CORE_PEER_ADDRESS=localho
st:7051
students@electro:~/fabric-samples/test-network$ Z
```

Using the Fabric test network

- Interactuando con la red



Se ejecuta el siguiente comando para inicializar el libro mayor con activos.

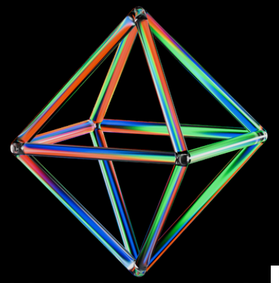
```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${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":[]}'
```

students@electro: ~/fabric-samples/test-network

```
students@electro:~/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${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":[]}'

Command 'peer' not found, did you mean:

  command 'beer' from deb gerstensaft
  command 'pear' from deb php-pear
  command 'peet' from deb pipexec
  command 'seer' from deb seer
  command 'pee' from deb moreutils

Try: sudo apt install <deb name>
```

```
-> INFO 001 Chaincode invoke successful. result: status:200
```

Using the Fabric test network



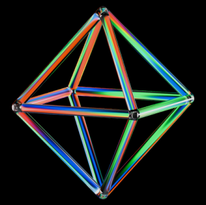
- Interactuando con la red

Se ejecuta el siguiente comando para obtener la lista de activos que se agregaron al registro de su canal: `peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'`

```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c
 '{"Args":["GetAllAssets"]}'

Command 'peer' not found, did you mean:

  command 'seer' from deb seer
  command 'pear' from deb php-pear
  command 'beer' from deb gerstensaft
  command 'pee' from deb moreutils
  command 'peet' from deb pipexec

Try: sudo apt install <deb name>
```

```
[
  {"ID": "asset1", "color": "blue", "size": 5, "owner": "Tomoko", "appraisedValue": 300},
  {"ID": "asset2", "color": "red", "size": 5, "owner": "Brad", "appraisedValue": 400},
  {"ID": "asset3", "color": "green", "size": 10, "owner": "Jin Soo", "appraisedValue": 500},
  {"ID": "asset4", "color": "yellow", "size": 10, "owner": "Max", "appraisedValue": 600},
  {"ID": "asset5", "color": "black", "size": 15, "owner": "Adriana", "appraisedValue": 700},
  {"ID": "asset6", "color": "white", "size": 15, "owner": "Michel", "appraisedValue": 800}
]
```

Using the Fabric test network



- Interactuando con la red

Se utiliza el siguiente comando para cambiar el propietario de un activo en el libro mayor invocando el código de cadena (básico) de transferencia de activos:

```
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":"TransferAsset","Args":["asset6","Christopher"]}'
```

students@electro: ~/fabric-samples/test-network

```
students@electro:~/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${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":"TransferAsset","Args":["asset6","Christopher"]}'

Command 'peer' not found, did you mean:

  command 'pee' from deb moreutils
  command 'beer' from deb gerstensaft
  command 'seer' from deb seer
  command 'pear' from deb php-pear
  command 'peet' from deb pipexec

Try: sudo apt install <deb name>
```

Si el comando es exitoso, debería ver la siguiente respuesta:

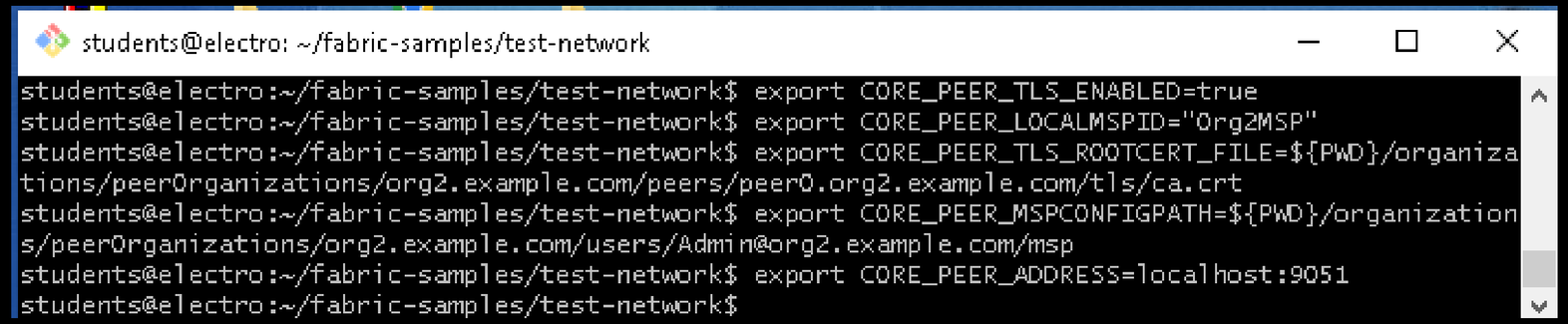
```
2019-12-04 17:38:21.048 EST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
```

Using the Fabric test network

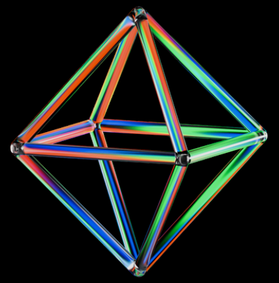
- Interactuando con la red

Como ya consultamos al par Org1, podemos aprovechar esta oportunidad para consultar el código de cadena que se ejecuta en el par Org2. Establezca las siguientes variables de entorno para operar como Org2:

```
# Environment variables for Org2
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051
```



Using the Fabric test network



- Interactuando con la red

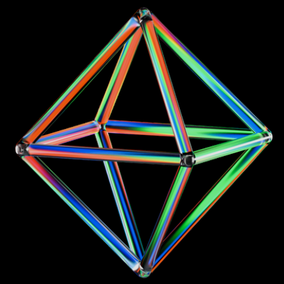
Ahora puede consultar el código de cadena de fabcar que se ejecuta en **peer0.org2.example.com**

peer chaincode query -C mychannel -n fabcar -c '{"Args":["queryCar","CAR9"]}'

```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ export CORE_PEER_ADDRESS=localhost:9051
students@electro:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c '{"A
rgs":["ReadAsset","asset6"]}'

Command 'peer' not found, did you mean:

  command 'pear' from deb php-pear
  command 'peet' from deb pipexec
  command 'beer' from deb gerstensaft
  command 'pee' from deb moreutils
  command 'seer' from deb seer

Try: sudo apt install <deb name>
```

```
{"ID":"asset6","color":"white","size":15,"owner":"Christopher","appraisedValue":800}
```

Using the Fabric test network



-Abrir la red con las autoridades de certificación

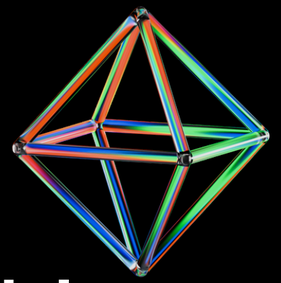
Si desea activar una red mediante Fabric CA, primero ejecute el siguiente comando para desactivar las redes en ejecución: `./network.sh down`

A continuación, puede abrir la red con la bandera CA: `./network.sh up -ca`

```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ ./network.sh up -ca
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and
db' with crypto from 'Certificate Authorities'
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
CA_LOCAL_VERSION=1.5.5
CA_DOCKER_IMAGE_VERSION=1.5.5
Generating certificates using Fabric CA
[+] Running 3/3
 ⠿ Container ca_org1      Started
 ⠿ Container ca_org2      Started
 ⠿ Container ca_orderer   Started
Creating Org1 Identities
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca
/home/students/fabric-samples/test-network/organizations/fabric-ca/org1/ca-ce
2022/08/29 22:27:19 [INFO] Created a default configuration file at /home/stud
st-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-
2022/08/29 22:27:19 [INFO] TLS Enabled
```

```
students@electro: ~/fabric-samples/test-network
ag to clean it up.
[+] Running 7/7
 ⠿ Volume "compose_peer0.org1.example.com"  Crea...                    0.0s
 ⠿ Volume "compose_peer0.org2.example.com"  Crea...                    0.0s
 ⠿ Volume "compose_orderer.example.com"     Created                    0.0s
 ⠿ Container peer0.org2.example.com         Started                    7.7s
 ⠿ Container orderer.example.com            Started                    7.7s
 ⠿ Container peer0.org1.example.com         Started                    7.1s
 ⠿ Container cli                            Started                    4.5s
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS
               PORTS                                                                        NAMES
f6a83651e5d8   hyperledger/fabric-tools:latest     "/bin/bash"              5 seconds ago    Up Les
s than a second                                                                             cli
3d9fcd2f9ba3   hyperledger/fabric-peer:latest      "peer node start"        9 seconds ago    Up 1 s
econd          0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9
445->9445/tcp                                                                               peer0.org2.example.com
7ca58fafe2ce   hyperledger/fabric-orderer:latest   "orderer"                9 seconds ago    Up 1 s
econd          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
4c179560590f   hyperledger/fabric-peer:latest      "peer node start"        9 seconds ago    Up 1 s
econd          0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/
tcp                                                                                         peer0.org1.example.com
098cacc6faaf   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   19 seconds ago   Up 15
seconds        0.0.0.0:8054->8054/tcp, :::8054->8054/tcp, 7054/tcp, 0.0.0.0:18054->18054/tcp, ::
:18054->18054/tcp                                                                           ca_org2
eb61feaf7f59   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   19 seconds ago   Up 16
seconds        0.0.0.0:9054->9054/tcp, :::9054->9054/tcp, 7054/tcp, 0.0.0.0:19054->19054/tcp, ::
:19054->19054/tcp                                                                           ca_orderer
34723bbca121   hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se…"   19 seconds ago   Up 15
seconds        0.0.0.0:7054->7054/tcp, :::7054->7054/tcp, 0.0.0.0:17054->17054/tcp, :::17054->17
054/tcp                                                                                     ca_org1
```

Using the Fabric test network



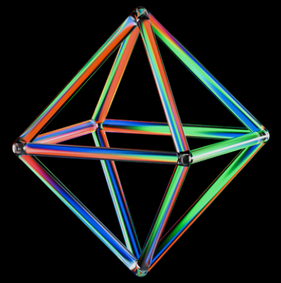
-Abrir la red con las autoridades de certificación

Se puede usar el siguiente comando para examinar la carpeta MSP del usuario administrador de Org1:

```
tree organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
```

```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ tree organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
└── msp
    ├── cacerts
    │   └── localhost-7054-ca-org1.pem
    ├── config.yaml
    ├── IssuerPublicKey
    ├── IssuerRevocationPublicKey
    ├── keystore
    │   └── 865669a2a4644e1bdbf86a055413b1a62f069a62094a4413002262abf0d7836d7_sk
    ├── signcerts
    │   └── cert.pem
    └── user
```

Using the Fabric test network

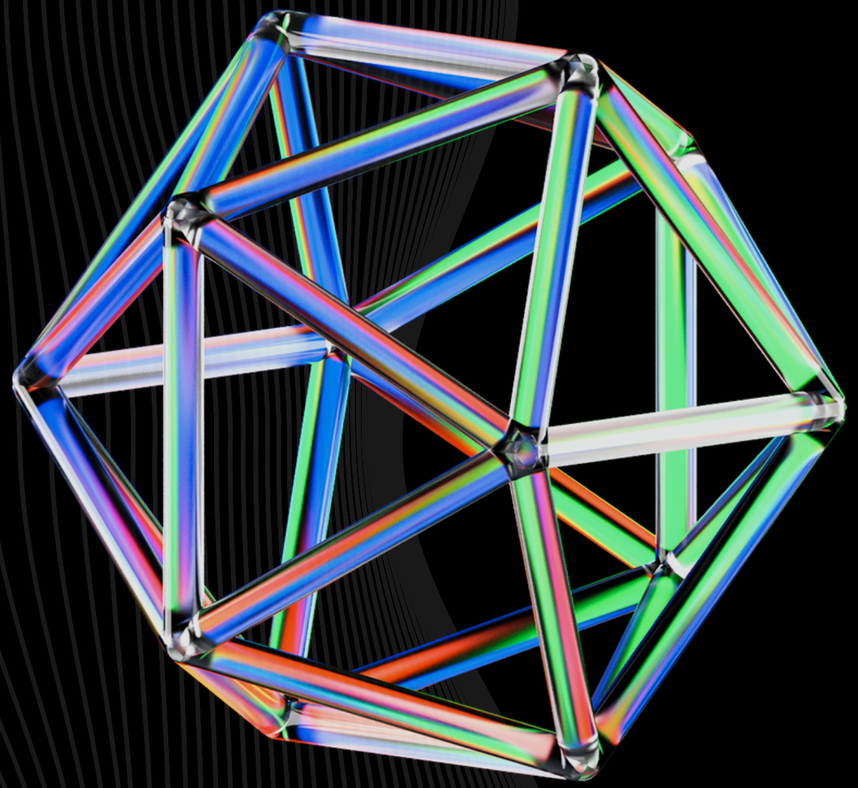


-Bajar la red

Cuando haya terminado de usar la red de prueba, puede desactivar la red con el siguiente comando:

./network.sh down

```
students@electro: ~/fabric-samples/test-network
students@electro:~/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
[+] Running 11/11
 ⠿ Container orderer.example.com          Removed                                2.9s
 ⠿ Container cli                          Removed                               11.3s
 ⠿ Container ca_orderer                   Removed                                2.1s
 ⠿ Container ca_org1                      Removed                                2.8s
 ⠿ Container ca_org2                      Removed                                2.6s
 ⠿ Container peer0.org2.example.com       Removed                                1.6s
 ⠿ Container peer0.org1.example.com       Removed                                1.6s
 ⠿ Volume compose_peer0.org3.example.com  Removed                                0.0s
 ⠿ Volume compose_orderer.example.com     Removed                                0.0s
 ⠿ Volume compose_peer0.org1.example.com  Removed                                0.1s
 ⠿ Volume compose_peer0.org2.example.com  Removed                                0.0s
Error: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com
Removing remaining containers
Removing generated chaincode docker images
"docker kill" requires at least 1 argument.
See 'docker kill --help'.

Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers
```



¡GRACIAS!