

RÉPUBLIQUE DU SÉNÉGAL



ÉCOLE SUPÉRIEURE POLYTECHNIQUE

DÉPARTEMENT GÉNIE INFORMATIQUE

**Module : Ingénierie de la cryptographie et Technologie
Blockchain**

Présenté par :

Serigne Mansor Dieng
Oumou Hawa Diallo
Mouhamadou Moustapha Sy
Cheikh Mbacké Ba

Professeur :

Dr. Mendy

Année universitaire 2023-2024

1. Téléchargez des exemples Fabric, des images Docker et des fichiers binaires :

Un répertoire de travail est requis - par exemple, les développeurs Go utilisent le `$HOME/go/src/github.com/mansor` répertoire. Il s'agit d'une recommandation de la communauté Golang pour les projets Go.

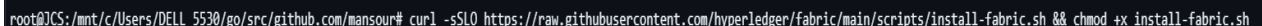
```
mkdir -p $HOME/go/src/github.com/mansor  
cd $HOME/go/src/github.com/mansor
```

2. Pour obtenir le script d'installation :

Pour obtenir le script d'installation nous allons exécuter la commande suivante :

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

Voici La description de la capture :



```
root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour# curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh
```

3. Choisir quels composants :

Pour spécifier les composants à télécharger, ajoutez un ou plusieurs des arguments suivants. Chaque argument peut être raccourci à sa première lettre.

- **docker** : utiliser Docker pour télécharger les images du conteneur Fabric
- **podman**: utiliser podman pour télécharger les images du conteneur Fabric
- **binary**: pour télécharger les binaires Fabric
- **samples**: pour cloner le dépôt github fabric-samples dans le répertoire actuel

Pour extraire les conteneurs Docker et cloner le dépôt d'exemples, exécutez par exemple l'une de ces commandes :

```
curl -sSL https://bit.ly/2ysbOFE | bash -s
```

Voici La description de la capture :

```
root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour# curl -sSL https://bit.ly/2ysb0FE | bash -s
```

Clone hyperledger/fabric-samples repo

==> Changing directory to fabric-samples

fabric-samples v2.5.7 does not exist, defaulting to main. fabric-samples main branch is intended to work with recent versions of fabric.

Pull Hyperledger Fabric binaries

==> Downloading version 2.5.7 platform specific fabric binaries

==> Downloading: <https://github.com/hyperledger/fabric/releases/download/v2.5.7/hyperledger-fabric-linux-amd64-2.5.7.tar.gz>

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
0	0	0	0	0	--:--:--	--:--:--	0s
100	109M	100	109M	0	0	4198k	0 0:00:26 0:00:26 --:--:-- 4460k

==> Done.

==> Downloading version 1.5.10 platform specific fabric-ca-client binary

==> Downloading: <https://github.com/hyperledger/fabric-ca/releases/download/v1.5.10/hyperledger-fabric-ca-linux-amd64-1.5.10.tar.gz>

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
0	0	0	0	0	--:--:--	--:--:--	0
100	28.6M	100	28.6M	0	0	3958k	0 0:00:07 0:00:07 --:--:-- 4829k

==> Done.

Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer ccenv baseos

==> Pulling fabric Images

====> docker.io/hyperledger/fabric-peer:2.5.7

2.5.7: Pulling from hyperledger/fabric-peer

Digest: sha256:a3dad3b96a1593a150f11124e8a2bcae13fe80e080733442a87735a0709a74e5

Status: Image is up to date for hyperledger/fabric-peer:2.5.7

docker.io/hyperledger/fabric-peer:2.5.7

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview docker.io/hyperledger/fabric-peer:2.5.7](#)

====> docker.io/hyperledger/fabric-orderer:2.5.7

2.5.7: Pulling from hyperledger/fabric-orderer

Digest: sha256:e3933b9ae1d0d7e725f90c50f8d6acf89dc1b99291e2ba2571e722ecd04023a3

Status: Image is up to date for hyperledger/fabric-orderer:2.5.7

docker.io/hyperledger/fabric-orderer:2.5.7

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview docker.io/hyperledger/fabric-orderer:2.5.7](#)

====> docker.io/hyperledger/fabric-ccenv:2.5.7

2.5.7: Pulling from hyperledger/fabric-ccenv

Digest: sha256:e6b1c6b8b5fb1667b6a28c878d4a8a2c32d7cad46e52f7d47068ace37a24d8fa

Status: Image is up to date for hyperledger/fabric-ccenv:2.5.7

docker.io/hyperledger/fabric-ccenv:2.5.7

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview docker.io/hyperledger/fabric-ccenv:2.5.7](#)

====> docker.io/hyperledger/fabric-baseos:2.5.7

2.5.7: Pulling from hyperledger/fabric-baseos

Digest: sha256:d7841906740f4edbd38730866fb887c20e886e034a975ebc80f1b4db73c3a199

Status: Image is up to date for hyperledger/fabric-baseos:2.5.7

docker.io/hyperledger/fabric-baseos:2.5.7

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview docker.io/hyperledger/fabric-baseos:2.5.7](#)

==> Pulling fabric ca Image

====> docker.io/hyperledger/fabric-ca:1.5.10

1.5.10: Pulling from hyperledger/fabric-ca

Digest: sha256:09a67ee71cfdb2861475d37cfc822f00545dc6852a43a6326e608b5926da1b5

Status: Image is up to date for hyperledger/fabric-ca:1.5.10

docker.io/hyperledger/fabric-ca:1.5.10

What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview docker.io/hyperledger/fabric-ca:1.5.10](#)

==> List out hyperledger docker images

hyperledger/fabric-peer	2.5	4b70009a7773	4 weeks ago	141MB
hyperledger/fabric-peer	2.5.7	4b70009a7773	4 weeks ago	141MB
hyperledger/fabric-peer	latest	4b70009a7773	4 weeks ago	141MB
hyperledger/fabric-orderer	2.5	3209e74fbdbb	4 weeks ago	110MB
hyperledger/fabric-orderer	2.5.7	3209e74fbdbb	4 weeks ago	110MB
hyperledger/fabric-orderer	latest	3209e74fbdbb	4 weeks ago	110MB
hyperledger/fabric-ccenv	2.5	682214ab2caa	4 weeks ago	629MB
hyperledger/fabric-ccenv	2.5.7	682214ab2caa	4 weeks ago	629MB
hyperledger/fabric-ccenv	latest	682214ab2caa	4 weeks ago	629MB
hyperledger/fabric-baseos	2.5	f8ac867caa68	4 weeks ago	128MB
hyperledger/fabric-baseos	2.5.7	f8ac867caa68	4 weeks ago	128MB
hyperledger/fabric-baseos	latest	f8ac867caa68	4 weeks ago	128MB
hyperledger/fabric-ca	1.5	da516cafd70e	4 weeks ago	206MB
hyperledger/fabric-ca	1.5.10	da516cafd70e	4 weeks ago	206MB
hyperledger/fabric-ca	latest	da516cafd70e	4 weeks ago	206MB

4. Afficher le réseau de test :

Nous pouvons trouver les scripts pour faire apparaître le réseau dans le **test-network** répertoire du **fabric-samples** référentiel. Accédez au répertoire du réseau de test à l'aide de la commande suivante :

```
cd fabric-samples/test-network
```

Voici La description de la capture :

```
root@ICS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour# cd fabric-samples/test-network
```

Dans ce répertoire, nous pouvons trouver un script annoté, **network.sh**, qui met en place un réseau Fabric à l'aide des images Docker sur votre ordinateur local. Vous pouvez exécuter pour imprimer le texte d'aide du script : **./network.sh -h**

Voici La description de la capture :

```

root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# ./network.sh -h
./network.sh: line 26: pushd: too many arguments
Using docker and docker-compose
Usage:
network.sh <Mode> [Flags]

Modes:
  prereq - Install Fabric binaries and docker images
  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-basic)
  cc - chaincode functions, use "network.sh cc -h" for options
  down - Bring down the network

Flags:
Used with network.sh prereq
-i    FabricVersion (default: '2.5.4')
-cai  Fabric CA Version (default: '1.5.7')

```

Used with network.sh up, network.sh createChannel:

```

-ca - Use Certificate Authorities to generate network crypto material
-cfssl <use CFSSL> - Use CFSSL CA to generate network crypto material
-bft - Use Orderers with consensus type BFT (Not available in Fabric v2.x)
-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 auto (default) or an integer, 1, 2, 3, etc
-ccp <path> - File path to the chaincode.
-cccp <policy> - (Optional) Chaincode endorsement policy using signature policy syntax. The default policy requires an endorsement from Org1 and Org2
-cccg <collection-config> - (Optional) File path to private data collections configuration file
-cci <fcn name> - (Optional) Name of chaincode initialization function. When a function is provided, the execution of init will be requested and the function will be invoked.
-h - Print this message

```

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

Examples:
network.sh up createChannel -ca -c mychannel -s couchdb
network.sh createChannel -c channelName
network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -ccl javascript
network.sh deployCC -ccn mychaincode -ccp ./user/mychaincode -ccv 1 -ccl javascript

NOTE: Default settings can be changed in network.config
./network.sh: line 1: popd: directory stack empty
```

5. Nous pouvons ensuite ouvrir le réseau en exécutant la commande suivante. Vous rencontrerez des problèmes si vous essayez d'exécuter le script depuis un autre répertoire :

En utilisant la commande suivante :

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

Cette commande crée un réseau Fabric composé de deux nœuds homologues et d'un nœud de classement. Aucun canal n'est créé lorsque vous exécutez, même si nous y reviendrons dans une [étape ultérieure](#) . Si la commande se termine avec succès, vous verrez les journaux des nœuds en cours de création : `./network.sh up`

Voici La description de la capture :

```
root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# ./network.sh up
```

```
./network.sh: line 26: pushd: too many arguments
```

```
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=v2.5.7
```

```
DOCKER_IMAGE_VERSION=v2.5.7
```

```
/mnt/c/Users/DELL 5530/go/src/github.com/mansour/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
```

```
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/compose-test-net.yaml: 'version' is obsolete
```

```
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/docker/docker-compose-test-net.yaml: 'version' is obsolete
```

```
[+] Running 7/7
```

```
✓ Network fabric_test Created 0.0s
```

```
✓ 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 orderer.example.com Started 1.3s
```

```
✓ Container peer0.org2.example.com Started 1.0s
```

```
✓ Container peer0.org1.example.com Started 1.2s
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
ded679d762ac	hyperledger/fabric-peer:latest	"peer node start"	2 seconds ago	Up Less than a second	0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp
7b7020dd64e3	hyperledger/fabric-peer:latest	"peer node start"	2 seconds ago	Up Less than a second	0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp
8273893ab100	hyperledger/fabric-orderer:latest	"orderer"	2 seconds ago	Up Less than a second	0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp

```
./network.sh: line 1: popd: directory stack empty
```

6. Les composants du réseau de test :

Une fois votre réseau de test déployé, vous pouvez prendre le temps d'examiner ses composants. Exécutez la commande suivante pour répertorier tous les conteneurs Docker en cours d'exécution sur votre ordinateur. Vous devriez voir les trois nœuds créés par le **network.sh** script :

En utilisant la commande suivante :

- **docker ps -a**

Voici La description de la capture :

```
root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
ded679d762ac	hyperledger/fabric-peer:latest	"peer node start"	About a minute ago	Up About a minute	0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp
7b7020dd64e3	hyperledger/fabric-peer:latest	"peer node start"	About a minute ago	Up About a minute	0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp
8273893ab100	hyperledger/fabric-orderer:latest	"orderer"	About a minute ago	Up About a minute	0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp

```
root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# ./network.sh createChannel
./network.sh: line 26: pushd: too many arguments
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'
Bringing up network
LOCAL_VERSION=v2.5.7
DOCKER_IMAGE_VERSION=v2.5.7
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/compose-test-net.yaml: `version` is obsolete
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/docker/docker-compose-test-net.yaml: `version` is obsolete
[+] Running 3/0
 ✓ Container peer0.org2.example.com Running 0.0s
 ✓ Container orderer.example.com Running 0.0s
 ✓ Container peer0.org1.example.com Running 0.0s
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAME
ded679d762ac	hyperledger/fabric-peer:latest	"peer node start"	2 minutes ago	Up 2 minutes	0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp	peer0
7b7020dd64e3	hyperledger/fabric-peer:latest	"peer node start"	2 minutes ago	Up 2 minutes	0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp	peer0
8273893ab100	hyperledger/fabric-orderer:latest	"orderer"	2 minutes ago	Up 2 minutes	0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp	order

```
/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/./bin/configtxgen
+ '[' 0 -eq 1 ']'
+ configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2024-05-17 17:23:41.842 GMT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2024-05-17 17:23:41.862 GMT 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2024-05-17 17:23:41.862 GMT 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms" election_tick:10
2024-05-17 17:23:41.862 GMT 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/
2024-05-17 17:23:42.095 GMT 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2024-05-17 17:23:42.095 GMT 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2024-05-17 17:23:42.096 GMT 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
environment: println: command not found
environment: println: command not found
+ . scripts/orderer.sh mychannel
+ '[' 0 -eq 1 ']'
+ res=0
Status: 201
{
  "name": "mychannel",
  "url": "/participation/v1/channels/mychannel",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
```



```

+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2024-05-17 17:23:50.928 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-05-17 17:23:50.961 GMT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
environment: println: command not found
environment: println: command not found
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2024-05-17 17:23:55.928 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-05-17 17:23:55.988 GMT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
environment: println: command not found
/mnt/c/Users/DELL: line 1: syntax error near unexpected token `('
/mnt/c/Users/DELL: line 1: `[161C:2A28][2024-02-07T11:28:36]i001: Burn v3.10.4.4718, Windows v10.0 (Build 22621: Service Pack 0), path: C:\Windows\Temp\{DAB8AAD26-0CE1-4538-BBFE-CD0087
environment: println: command not found
environment: println: command not found
scripts/setAnchorPeer.sh: line 21: fetchChannelConfig: command not found
environment: println: command not found
++ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1.example.com","port": 7051}]}},'v
scripts/setAnchorPeer.sh: line 40: ${TEST_NETWORK_HOME}/channel-artifacts/${CORE_PEER_LOCALMSPID}modified_config.json: ambiguous redirect
++ res=1

```

7. Créer une chaîne :

Maintenant que nous avons des nœuds homologues et trieurs exécutés sur notre machine, nous pouvons utiliser le script pour créer un canal Fabric pour les transactions entre Org1 et Org2. Les canaux constituent une couche privée de communication entre des membres spécifiques du réseau. Les chaînes ne peuvent être utilisées que par les organisations invitées sur la chaîne et sont invisibles pour les autres membres du réseau. Chaque canal dispose d'un registre blockchain distinct. Les organisations qui ont été invitées « rejoignent » leurs pairs sur le canal pour stocker le grand livre du canal et valider les transactions sur le canal.

Vous pouvez utiliser le **network.sh** script pour créer un canal entre Org1 et Org2 et joindre leurs pairs au canal. Exécutez la commande suivante pour créer un canal avec le nom par défaut **mychannel** :

En utilisant la commande suivante :

- **./network.sh createChannel**

Voici La description de la capture :

```

root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# ./network.sh createChannel

./network.sh: line 26: pushd: too many arguments
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'
Bringing up network
LOCAL_VERSION=v2.5.7
DOCKER_IMAGE_VERSION=v2.5.7
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/compose-test-net.yaml: 'version' is obsolete
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/docker/docker-compose-test-net.yaml: 'version' is obsolete
[*] Running 3/0
✓ Container peer0.org2.example.com Running 0.0s
✓ Container orderer.example.com Running 0.0s
✓ Container peer0.org1.example.com Running 0.0s

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ded679d762ac	hyperledger/fabric-peer:latest	"peer node start"	5 minutes ago	Up 5 minutes	0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp	peer0.or
7b7020dd64e3	hyperledger/fabric-peer:latest	"peer node start"	5 minutes ago	Up 5 minutes	0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp	peer0.or
8273893ab100	hyperledger/fabric-orderer:latest	"orderer"	5 minutes ago	Up 5 minutes	0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp	orderer.

```

/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/./bin/configtxgen
+ '[' 0 -eq 1 ']'
+ configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2024-05-17 17:26:04.937 GMT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2024-05-17 17:26:04.959 GMT 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2024-05-17 17:26:04.959 GMT 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms" election_tick:1
2024-05-17 17:26:04.959 GMT 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network
2024-05-17 17:26:05.188 GMT 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2024-05-17 17:26:05.189 GMT 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2024-05-17 17:26:05.189 GMT 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
environment: println: command not found
environment: println: command not found
+ . scripts/orderer.sh mychannel
+ '[' 0 -eq 1 ']'
+ res=0
2024-05-17 17:24:00.171 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
Status: 405
{
  "error": "cannot join: channel already exists"
}

```

```

+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
2024-05-17 17:26:28.856 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: proposal failed (err: bad proposal response 500: cannot create ledger from genesis block: ledger [mychannel] already exists with state [ACTIVE])
scripts/envVar.sh: line 87: fatalln: command not found
environment: println: command not found
environment: println: command not found
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=1
2024-05-17 17:26:48.802 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: proposal failed (err: bad proposal response 500: cannot create ledger from genesis block: ledger [mychannel] already exists with state [ACTIVE])

```

```

++ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1.example.com","port": 7051}]}},'
scripts/setAnchorPeer.sh: line 40: ${TEST_NETWORK_HOME}/channel-artifacts/${CORE_PEER_LOCALMSPID}modified_config.json: ambiguous redirect
++ res=1
scripts/envVar.sh: line 87: fatalln: command not found
scripts/setAnchorPeer.sh: line 49: createConfigUpdate: command not found
2024-05-17 17:26:50.923 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
scripts/envVar.sh: line 87: fatalln: command not found
environment: println: command not found
environment: println: command not found
/mnt/c/Users/DELL: line 1: syntax error near unexpected token `('
/mnt/c/Users/DELL: line 1: `[161C:2A28][2024-02-07T11:28:36]1001: Burn v3.10.4.4718, Windows v10.0 (Build 22621: Service Pack 0), path: C:\Windows\Temp\{DABAA26-0CE1-4538-BBFE-CDD087
environment: println: command not found
environment: println: command not found
scripts/setAnchorPeer.sh: line 21: fetchChannelConfig: command not found
environment: println: command not found
++ jq '.channel_group.groups.Application.groups.Org2MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org2.example.com","port": 9051}]}},'
scripts/setAnchorPeer.sh: line 40: ${TEST_NETWORK_HOME}/channel-artifacts/${CORE_PEER_LOCALMSPID}modified_config.json: ambiguous redirect
++ res=1
scripts/envVar.sh: line 87: fatalln: command not found
scripts/setAnchorPeer.sh: line 49: createConfigUpdate: command not found
2024-05-17 17:26:53.029 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
scripts/envVar.sh: line 87: fatalln: command not found
environment: println: command not found
environment: println: command not found
./network.sh: line 1: popd: directory stack empty

```

Nous pouvons également utiliser l'indicateur de chaîne pour créer une chaîne avec un nom personnalisé. A titre d'exemple, la commande suivante créerait un canal nommé **channel1** :

En utilisant la commande suivante :

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

Voici La description de la capture :

```

root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# ./network.sh createChannel -c channel1

./network.sh: line 26: pushd: too many arguments
Using docker and docker-compose
Creating channel 'channel1'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
Bringing up network
LOCAL_VERSION=v2.5.7
DOCKER_IMAGE_VERSION=v2.5.7
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/compose-test-net.yaml: 'version' is obsolete
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/docker/docker-compose-test-net.yaml: 'version' is obsolete
[+] Running 3/0
 ✓ Container orderer.example.com    Running    0.0s
 ✓ Container peer0.org1.example.com  Running    0.0s
 ✓ Container peer0.org2.example.com  Running    0.0s
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                                                                                                     NAMES
ded679d762ac   hyperledger/fabric-peer:latest      "peer node start"       6 minutes ago Up 6 minutes   0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp   peer0.org1
7b7020dd64e3   hyperledger/fabric-peer:latest      "peer node start"       6 minutes ago Up 6 minutes   0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp             peer0.org2
8273893ab100   hyperledger/fabric-orderer:latest   "orderer"               6 minutes ago Up 6 minutes   0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp   orderer.example.com

```

```

/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/bin/configtxgen
+ '[' 0 -eq 1 ']'
+ configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/channel1.block -channelID channel1
2024-05-17 17:27:25.379 GMT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2024-05-17 17:27:25.402 GMT 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2024-05-17 17:27:25.402 GMT 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms" election_tick:10
2024-05-17 17:27:25.402 GMT 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/
2024-05-17 17:27:25.625 GMT 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2024-05-17 17:27:25.625 GMT 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2024-05-17 17:27:25.626 GMT 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
environment: println: command not found
environment: println: command not found
+ . scripts/orderer.sh channel1
+ '[' 0 -eq 1 ']'
+ res=0
2024-05-17 17:26:53.029 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
Status: 201
{
  "name": "channel1",
  "url": "/participation/v1/channels/channel1",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}

```

```

+ peer channel join -b ./channel-artifacts/channel1.block
+ res=0
2024-05-17 17:27:34.370 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-05-17 17:27:34.401 GMT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
environment: println: command not found
environment: println: command not found
+ peer channel join -b ./channel-artifacts/channel1.block
+ res=0
2024-05-17 17:27:39.383 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-05-17 17:27:39.422 GMT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
environment: println: command not found
/mnt/c/Users/DELL: line 1: syntax error near unexpected token `('
/mnt/c/Users/DELL: line 1: `[161C:2A28][2024-02-07T11:28:36]i001: Burn v3.10.4.4718, Windows v10.0 (Build 22621: Service Pack 0), path: C:\Windows\Temp\{DA8AA026-0CE1-4538-BBFE-CDD0887F
environment: println: command not found
environment: println: command not found
scripts/setAnchorPeer.sh: line 21: fetchChannelConfig: command not found
environment: println: command not found
++ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1.example.com","port": 7051}]}},'ve
scripts/setAnchorPeer.sh: line 40: ${TEST_NETWORK_HOME}/channel-artifacts/${CORE_PEER_LOCALMSPID}modified_config.json: ambiguous redirect
++ res=1

```

```

2024-05-17 17:27:41.389 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
scripts/envVar.sh: line 87: fatalln: command not found
environment: println: command not found
environment: println: command not found
/mnt/c/Users/DELL: line 1: syntax error near unexpected token `('
/mnt/c/Users/DELL: line 1: `[161C:2A28][2024-02-07T11:28:36]i001: Burn v3.10.4.4718, Windows v10.0 (Build 22621: Service Pack 0), path: C:\Windows\Temp\{D48AAD26-0CE1-4538-BBFE-CD
environment: println: command not found
environment: println: command not found
scripts/setAnchorPeer.sh: line 21: fetchChannelConfig: command not found
environment: println: command not found
++ jq '.channel_group.groups.Application.groups.Org2MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org2.example.com","port": 9051}]
scripts/setAnchorPeer.sh: line 40: ${TEST_NETWORK_HOME}/channel-artifacts/${CORE_PEER_LOCALMSPID}modified_config.json: ambiguous redirect
++ res=1
scripts/envVar.sh: line 87: fatalln: command not found
scripts/setAnchorPeer.sh: line 49: createConfigUpdate: command not found
2024-05-17 17:27:43.695 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
scripts/envVar.sh: line 87: fatalln: command not found
environment: println: command not found
environment: println: command not found
./network.sh: line 1: popd: directory stack empty

```

L'indicateur de canal nous permet également de créer plusieurs canaux en spécifiant différents noms de canal. Après avoir créé **mychannel** ou **channel1**, vous pouvez utiliser la commande ci-dessous pour créer un deuxième canal nommé **channel2** :

En utilisant la commande suivante :

- `./network.sh createChannel -c channel2`

Voici La description de la capture :

```
root@ICS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# ./network.sh createChannel -c channel2
```

```
./network.sh: line 26: pushd: too many arguments
```

```
Using docker and docker-compose
```

```
Creating channel 'channel2'.
```

```
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
```

```
Bringing up network
```

```
LOCAL_VERSION=v2.5.7
```

```
DOCKER_IMAGE_VERSION=v2.5.7
```

```
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/compose-test-net.yaml: `version` is obsolete
```

```
WARN[0000] /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/compose/docker/docker-compose-test-net.yaml: `version` is obsolete
```

```
[+] Running 3/0
```

```
✓ Container peer0.org2.example.com Running 0.0s
```

```
✓ Container orderer.example.com Running 0.0s
```

```
✓ Container peer0.org1.example.com Running 0.0s
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ded679d762ac	hyperledger/fabric-peer:latest	"peer node start"	7 minutes ago	Up 7 minutes	0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp	peer0.org2
7b7020dd64e3	hyperledger/fabric-peer:latest	"peer node start"	7 minutes ago	Up 7 minutes	0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp	peer0.org1
8273893ab100	hyperledger/fabric-orderer:latest	"orderer"	7 minutes ago	Up 7 minutes	0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp	orderer.example.com

```
/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/./bin/configtxgen
```

```
+ '[' 0 -eq 1 ']'
```

```
+ configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/channel2.block -channelID channel2
```

```
2024-05-17 17:28:05.992 GMT 0001 INFO [common.tools.configtxgen] main -> Loading configuration
```

```
2024-05-17 17:28:06.007 GMT 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
```

```
2024-05-17 17:28:06.007 GMT 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms" election_tick:10
```

```
2024-05-17 17:28:06.007 GMT 0004 INFO [common.tools.configtxgen.localconfig] load -> Loaded configuration: /mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network/
```

```
2024-05-17 17:28:06.248 GMT 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
```

```
2024-05-17 17:28:06.248 GMT 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
```

```
2024-05-17 17:28:06.249 GMT 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
```

```
+ res=0
```

```
environment: println: command not found
```

```
environment: println: command not found
```

```
+ . scripts/orderer.sh channel2
```

```
+ '[' 0 -eq 1 ']'
```

```
+ res=0
```

```
2024-05-17 17:27:43.695 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
```

```
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
```

```
Status: 201
```

```
{
  "name": "channel2",
  "url": "/participation/v1/channels/channel2",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
```

```

+ peer channel join -b ./channel-artifacts/channel2.block
+ res=0
2024-05-17 17:28:14.988 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-05-17 17:28:15.039 GMT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
environment: println: command not found
environment: println: command not found
+ peer channel join -b ./channel-artifacts/channel2.block
+ res=0
2024-05-17 17:28:19.992 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-05-17 17:28:20.049 GMT 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
environment: println: command not found
/mnt/c/Users/DELL: line 1: syntax error near unexpected token '('
/mnt/c/Users/DELL: line 1: `[161C:2A28][2024-02-07T11:28:36]i001: Burn v3.10.4.4718, Windows v10.0 (Build 22621: Service Pack 0), path: C:\Windows\Temp\[DAB8AAD26-0CE1-4538-BBFE-CDD0887F
environment: println: command not found
environment: println: command not found
scripts/setAnchorPeer.sh: line 21: fetchChannelConfig: command not found
environment: println: command not found
++ jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1.example.com","port": 7051}]}}'
scripts/setAnchorPeer.sh: line 40: ${TEST_NETWORK_HOME}/channel-artifacts/${CORE_PEER_LOCALMSPID}modified_config.json: ambiguous redirect
++ res=1
scripts/envVar.sh: line 87: fatalln: command not found
scripts/setAnchorPeer.sh: line 49: createConfigUpdate: command not found
2024-05-17 17:28:22.097 GMT 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Error: error unmarshalling Envelope: proto: cannot parse invalid wire-format data
scripts/envVar.sh: line 87: fatalln: command not found
environment: println: command not found
environment: println: command not found
/mnt/c/Users/DELL: line 1: syntax error near unexpected token '('
/mnt/c/Users/DELL: line 1: `[161C:2A28][2024-02-07T11:28:36]i001: Burn v3.10.4.4718, Windows v10.0 (Build 22621: Service Pack 0), path: C:\Windows\Temp\[DAB8AAD26-0CE1-4538-BBFE-CD

```

8. Démarrer un chaincode sur la chaîne

Avant de conditionner le code chaîne, vous devez installer les dépendances du code chaîne.

D'abord, on crée le dossier contenant la version Go du code chaîne de transfert d'actifs (de base) à l'aide de la commande suivante:

- `mkdir -p $GOPATH/src/github.com/your_org/asset`

Voici La description de la capture :

```

root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# mkdir -p $GOPATH/src/github.com/your_org/asset

```

Ensuite, on se déplace sur le dossier qui va contenir la version Go du code chaîne de transfert d'actifs (de base) puis on crée notre fichier à l'aide des commandes suivantes:

- `cd $GOPATH/src/github.com/your_org/asset`
- `touch chaincode.go`

Voici La description de la capture :

```

root@JCS:/mnt/c/Users/DELL 5530/go/src/github.com/mansour/fabric-samples/test-network# cd $GOPATH/src/github.com/your_org/asset
root@JCS:/src/github.com/your_org/asset# touch chaincode.go

```

Après, on fait des configurations dans le fichier chaincode.go puis on visualise le contenu en tapant les commandes de façon respectives:

- `nano chaincode.go`
- `cat chaincode.go`

Voici La description de la capture :

```
root@JCS:/src/github.com/your_org/asset# nano chaincode.go
root@JCS:/src/github.com/your_org/asset# cat chaincode.go
package main

import (
    "encoding/json"
    "fmt"
    "github.com/hyperledger/fabric-contract-api-go/contractapi"
)

type AssetContract struct {
    contractapi.Contract
}

type Asset struct {
    ID    string `json:"id"`
    Value string `json:"value"`
}
```

```
func (c *AssetContract) CreateAsset(ctx contractapi.TransactionContextInterface, id string, value string) error {
    asset := Asset{
        ID:    id,
        Value: value,
    }

    assetJSON, err := json.Marshal(asset)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutPrivateData("collectionAssets", id, assetJSON)
}

func (c *AssetContract) ReadAsset(ctx contractapi.TransactionContextInterface, id string) (*Asset, error) {
    assetJSON, err := ctx.GetStub().GetPrivateData("collectionAssets", id)
    if err != nil {
        return nil, err
    }
    if assetJSON == nil {
        return nil, fmt.Errorf("asset not found: %s", id)
    }

    var asset Asset
    err = json.Unmarshal(assetJSON, &asset)
    if err != nil {
        return nil, err
    }

    return &asset, nil
}
```



```
func main() {
    chaincode, err := contractapi.NewChaincode(new(AssetContract))
    if err != nil {
        fmt.Printf("Error create asset chaincode: %s", err)
        return
    }

    if err := chaincode.Start(); err != nil {
        fmt.Printf("Error starting asset chaincode: %s", err)
    }
}
```

Nous pouvons utiliser la **peer** CLI pour créer un package de code chaîne au format requis. Les **peer** binaires se trouvent dans le **bin** dossier du **fabric-samples** référentiel. Utilisez les commande suivantes pour ajouter ces binaires à votre chemin CLI :

- `export PATH=$PATH:/path/to/fabric-samples/bin`
- `export PATH=$PATH:$HOME/fabric-samples/bin`

Voici La description de la capture :

```
root@JCS:/src/github.com/your_org/asset# export PATH=$PATH:/path/to/fabric-samples/bin
root@JCS:/src/github.com/your_org/asset# export PATH=$PATH:$HOME/fabric-samples/bin
```

Pour confirmer que vous pouvez utiliser la **peer** CLI, vérifiez la version des binaires en tapant la commande suivante:

- `peer version`

Les binaires doivent être en version **2.0.0** ou ultérieure pour exécuter ce didacticiel.

Voici La description de la capture :

```
root@JCS:/src/github.com/your_org/asset# peer version
peer:
Version: v2.5.8
Commit SHA: ab5290f
Go version: go1.22.3
OS/Arch: linux/amd64
Chaincode:
Base Docker Label: org.hyperledger.fabric
Docker Namespace: hyperledger
```

Vous pouvez maintenant créer le package chaincode à l'aide de la commande **peer lifecycle chaincode package** :

- `peer lifecycle chaincode package asset.tar.gz --path . --lang golang --label asset_1`

Cette commande créera un package nommé `basic.tar.gz` dans votre répertoire actuel. L' `--lang` indicateur est utilisé pour spécifier le langage du code de chaîne et l' `--path` indicateur fournit l'emplacement de votre code de contrat intelligent. Le chemin doit être un chemin complet ou un chemin relatif à votre répertoire de travail actuel. L' `--label` indicateur est utilisé pour spécifier une étiquette de code de chaîne qui identifiera votre code de chaîne après son installation. Il est recommandé que votre étiquette inclut le nom et la version du code de chaîne.

Voici La description de la capture :

```
root@JCS:/src/github.com/your_org/asset# peer lifecycle chaincode package asset.tar.gz --path . --lang golang --label asset_1
peer lifecycle chaincode install asset.tar.gz
peer lifecycle chaincode queryinstalled
```

Après avoir utilisé le `network.sh` pour créer une chaîne, et installer les dépendances du code chaîne(package chaincode), enfin, nous pouvons démarrer maintenant un chaincode sur la chaîne à l'aide de la commande suivante :

- `./network.sh deployCC -ccn basic -ccp /src/github.com/your_org/asset -ccl go`

Ce script utilise le cycle de vie du code de chaîne pour emballer, installer, interroger le code de chaîne installé, approuver le code de chaîne pour Org1 et Org2, et enfin valider le code de chaîne.

Voici La description de la capture :

```
root@JCS:/src/github.com/your_org# ./network.sh deployCC -ccn basic -ccp . -ccl go
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vsc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
```

Configuration d'une solution de stockage centralisée :

Pour configurer un espace de stockage centralisé, nous allons utiliser NFS (Network File System).

Installation de NFS sur le serveur :

```
sudo apt update
sudo apt install nfs-kernel-server
```

Création d'un répertoire à partager et modification des permissions :

```
sudo mkdir -p /mnt/nfs_share
sudo chown nobody:nogroup /mnt/nfs_share
sudo chmod 777 /mnt/nfs_share
```

Configuration des exports NFS :

Pour réaliser cette étape, il faut éditez le fichier `/etc/exports` et ajoutez la ligne suivante :

```
/mnt/nfs_share    *(rw,sync,no_subtree_check)
```

Démarrage du serveur NFS :

```
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

Configuration NFS sur le client :

```
#Installation des utilitaires NFS
sudo apt update
sudo apt install nfs-common

#Montage du partage NFS
sudo mount -t nfs 192.168.1.102:/mnt/nfs_share /mnt
```

Tests de conformité et de sécurité :

Utilisation de nmap pour scanner les ports ouverts :

Nmap, abréviation de Network Mapper, est un outil de scan de réseau largement utilisé pour découvrir les hôtes et services sur un réseau informatique, ainsi que pour analyser la sécurité des systèmes.

```
sudo apt install nmap
```

Une fois installé, nous pouvons exécuter un scan de ports sur un hôte en utilisant son adresse IP.

```
nmap -v -A 192.168.1.102
Starting Nmap 7.95 ( https://nmap.org ) at 2024-05-
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 12:44
Completed NSE at 12:44, 0.00s elapsed
Initiating NSE at 12:44
Completed NSE at 12:44, 0.00s elapsed
Initiating NSE at 12:44
Completed NSE at 12:44, 0.00s elapsed
Initiating ARP Ping Scan at 12:44
Scanning 192.168.1.102 [1 port]
Completed ARP Ping Scan at 12:44, 1.50s elapsed (1
Nmap scan report for 192.168.1.102 [host down]
NSE: Script Post-scanning.
Initiating NSE at 12:44
Completed NSE at 12:44, 0.00s elapsed
Initiating NSE at 12:44
Completed NSE at 12:44, 0.00s elapsed
Initiating NSE at 12:44
Completed NSE at 12:44, 0.01s elapsed
```

Utilisation de clamav pour vérifier les fichiers :

ClamAV est un logiciel antivirus open-source largement utilisé pour détecter les menaces sur les systèmes Linux.

```
sudo apt install clamav
sudo freshclam
sudo clamscan -r /mnt/nfs_share
```

Configuration du firewall avec ufw :

```
sudo apt install ufw
sudo ufw allow from 192.168.1.13 to any port nfs
sudo ufw allow from 192.168.1.13 to any port 445
sudo ufw enable
```

Utilisation de osquery pour auditer la conformité :

Pour effectuer des tests de conformité sur un système de stockage centralisé dans un contexte blockchain, il est crucial d'assurer que votre solution respecte les normes et les réglementations en vigueur. Voici les étapes détaillées pour effectuer des tests de conformité, y compris l'utilisation d'outils spécifiques pour vérifier les permissions, la sécurité des données, et la conformité réglementaire :

Avant de commencer les tests, il est important d'identifier les normes et réglementations applicables à votre système de stockage de documents. Voici quelques exemples de normes couramment appliquées :

- GDPR (General Data Protection Regulation) pour les données personnelles dans l'UE.
- HIPAA (Health Insurance Portability and Accountability Act) pour les informations de santé aux États-Unis.
- SOX (Sarbanes-Oxley Act) pour les pratiques comptables aux États-Unis.
- PCI-DSS (Payment Card Industry Data Security Standard) pour les informations de carte de crédit.

osquery est un outil open-source qui permet d'interroger les systèmes en utilisant des requêtes SQL pour vérifier les configurations et les permissions.

```
sudo apt install osquery
sudo osqueryi --line "SELECT * FROM compliance;"
```

Analyse des Coûts et Comparaison :

L'analyse des coûts et la comparaison entre les solutions de stockage centralisé (comme NFS ou SMB) et les solutions de stockage décentralisé (comme IPFS, Sia, Storj) impliquent plusieurs facteurs.

Coûts Initiaux :

Solution Centralisée (NFS/SMB) :

- **Matériel** : Serveur(s) de stockage, disques durs, équipement réseau.
 - Exemple : Un serveur NAS avec 10 To de stockage peut coûter entre 500 et 2000 euros selon la marque et les spécifications.
- **Logiciel** : Coût de licence si vous utilisez un logiciel commercial, bien que NFS et Samba soient open-source et gratuits.
 - NFS/Samba : Généralement gratuit, inclus avec la plupart des distributions Linux.

Solution Décentralisée (IPFS, Sia, Storj) :

- **Matériel** : Minimal, souvent juste un nœud léger pour interfacer avec le réseau décentralisé.
 - Exemple : Un ordinateur ou un serveur léger pour exécuter les clients IPFS/Sia/Storj.
- **Logiciel** : Généralement open-source et gratuit.
 - IPFS : Gratuit.
 - Sia : Frais de transaction mineurs pour stocker des fichiers.
 - Storj : Pay-per-use model avec des frais de stockage et de bande passante.

Coûts Opérationnels :

Solution Centralisée (NFS/SMB) :

- **Énergie** : Consommation électrique des serveurs et du matériel réseau.
 - Exemple : Un serveur NAS consomme entre 50W et 200W, ce qui peut représenter entre 32800 et 131300 FCFA par an en électricité.
- **Maintenance** : Coûts de maintenance du matériel, mises à jour logicielles, sauvegardes.
 - Exemple : Coûts de remplacement de disques défectueux, frais de support technique.
- **Personnel** : Coût du personnel pour gérer et maintenir l'infrastructure.
 - Exemple : Salaire d'un administrateur système.

Solution Décentralisée (IPFS, Sia, Storj) :

- **Frais de stockage** : Pay-as-you-go pour le stockage des données.
 - Sia : Environ 1200 à 3000 FCFA par mois pour 1 To.
 - Storj : Environ 2400 FCFA par mois pour 1 To, avec des frais supplémentaires pour la bande passante.
- **Bande passante** : Frais liés à l'upload et au download de données.
 - Exemple : Storj facture environ 4250 FCFA par mois pour 1 To de bande passante sortante.
- **Résilience** : Moins de coûts de maintenance de matériel, mais des coûts pour assurer la disponibilité et la redondance.

Sécurité et Résilience :

Solution Centralisée (NFS/SMB) :

- **Sécurité** : Dépend de la configuration et de la maintenance du serveur.
 - Exemple : Besoin de mesures de sécurité robustes (firewall, antivirus, etc.).
- **Résilience** : Doit inclure des plans de sauvegarde et de récupération en cas de panne.
 - Exemple : RAID pour la redondance des disques, sauvegardes régulières.

Solution Décentralisée (IPFS, Sia, Storj) :

- **Sécurité** : Les données sont chiffrées et distribuées, augmentant la sécurité contre les attaques centralisées.
 - Exemple : Sia chiffre les fichiers avant de les distribuer sur le réseau.
- **Résilience** : Haute résilience grâce à la distribution des données sur plusieurs nœuds.
 - Exemple : IPFS assure la disponibilité en répliquant les données sur plusieurs nœuds.

Conformité réglementaire:

Solution Centralisée (NFS/SMB) :

- **Contrôle Total** : Contrôle complet sur l'infrastructure, facilitant la conformité.
 - Exemple : Plus facile à auditer et à certifier pour des réglementations spécifiques comme GDPR, HIPAA.
- **Données Localisées** : Les données peuvent être stockées dans des régions spécifiques pour répondre aux exigences de souveraineté des données.

Solution Décentralisée (IPFS, Sia, Storj) :

- **Complexité Réglementaire** : Difficulté à garantir la conformité en raison de la nature distribuée.
 - Exemple : Difficulté à auditer les nœuds individuels.
- **Données Réparties** : La distribution globale peut compliquer la conformité avec certaines réglementations locales.
 - Exemple : Défis pour garantir que les données restent dans une juridiction spécifique.

Conclusion :

Avantages des Solutions Centralisées (NFS/SMB) :

- Contrôle total sur les données et l'infrastructure.
- Facilité de conformité réglementaire.
- Sécurité et résilience personnalisables.

Inconvénients des Solutions Centralisées (NFS/SMB) :

- Coûts initiaux et opérationnels élevés.
- Limites physiques et scalabilité réduite.
- Maintenance continue nécessaire.

Avantages des Solutions Décentralisées (IPFS, Sia, Storj) :

- Scalabilité élastique et illimitée.
- Coûts potentiellement plus faibles pour le stockage à grande échelle.
- Haute sécurité et résilience intrinsèques.

Inconvénients des Solutions Décentralisées (IPFS, Sia, Storj) :

- Complexité de conformité réglementaire.
- Dépendance à l'infrastructure réseau décentralisée.

- Moins de contrôle direct sur les données.

Ce projet a permis de mettre en lumière les forces et les faiblesses des différentes approches de stockage dans un contexte blockchain. Les solutions centralisées offrent un contrôle et une conformité plus faciles, mais à des coûts plus élevés et avec des limitations de scalabilité. En revanche, les solutions décentralisées apportent une flexibilité et une résilience accrues, bien qu'elles présentent des défis en termes de conformité et de gestion des données.

En conclusion, le choix entre une solution de stockage centralisée ou décentralisée dépend largement des besoins spécifiques de l'application, du budget disponible, et des exigences réglementaires.