

Hyperledyer Fabric: Manual de instalación

René Dávila - Jorge Solano

Manual de instalación de Hyperledger Fabric.

Prerequisitos

1. Instalar git:
<https://git-scm.com/downloads>.
2. Instalar cURL:
<https://curl.haxx.se/download.html>.
3. Instalar Docker y Docker Compose (Se requiere la versión 17.06.2-ce o superior):
<https://www.docker.com/get-docker>.
4. Instalar Go:
<https://golang.org/dl/>.
5. Instalar Node.js y NPM: <https://nodejs.org/en/download/>.
6. Instalar Python: <https://www.python.org/downloads/>.

Para construir el escenario de la primera red en Hyperledger Fabric es necesario, después de instalar los prerequisites, instalar los ejemplos, binarios e imágenes Docker que se encuentran en <https://hyperledger-fabric.readthedocs.io/en/latest/install.html>. Para ello hay que iniciar Docker Desktop y ejecutar la línea instrucción:

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

La instrucción anterior descarga la carpeta *fabric-samples* en la ubicación actual

```
└─ fabric-samples
```

La fábrica de ejemplos descarga varios proyectos con diferentes escenarios.

```
.
├── CODEOWNERS
├── CODE_OF_CONDUCT.md
├── CONTRIBUTING.md
├── Jenkinsfile
├── LICENSE
├── MAINTAINERS.md
├── README.md
├── SECURITY.md
├── bin
├── chaincode
├── chaincode-docker-devmode
├── ci
├── ci.properties
├── commercial-paper
├── config
├── docs
├── fabcar
├── first-network
├── high-throughput
├── interest_rate_swaps
├── off_chain_data
├── scripts
└── test-network
```

Para iniciar se recomienda ejecutar el ejemplo test-network, el cual permite ejecutar nodos en la máquina local. Este proyecto también permite probar aplicaciones y contratos inteligentes.

```
.
├── CODEOWNERS
├── CODE_OF_CONDUCT.md
├── CONTRIBUTING.md
├── Jenkinsfile
├── LICENSE
├── MAINTAINERS.md
├── README.md
├── SECURITY.md
├── bin
├── chaincode
├── chaincode-docker-devmode
├── ci
├── ci.properties
├── commercial-paper
├── config
├── docs
├── fabcar
├── first-network
├── high-throughput
├── interest_rate_swaps
├── off_chain_data
├── scripts
└── test-network
```

Si se ejecuta el script byfn.sh (build your first network) sin argumentos, como salida se obtiene una descripción de diversas opciones para ejecutar el ejemplo.

```

Usage:
  byfn.sh <mode> [-c <channel name>] [-t <timeout>] [-d <delay>] [-f <docker-compose-file>] [-s <dbtype>] [-l <language>] [-o <consensus-type>] [-i <imagetag>] [-a] [-n] [-v]
  <mode> - one of 'up', 'down', 'restart', 'generate' or 'upgrade'
    - 'up' - bring up the network with docker-compose up
    - 'down' - clear the network with docker-compose down
    - 'restart' - restart the network
    - 'generate' - generate required certificates and genesis block
    - 'upgrade' - upgrade the network from version 1.3.x to 1.4.0
  -c <channel name> - channel name to use (defaults to "mychannel")
  -t <timeout> - CLI timeout duration in seconds (defaults to 10)
  -d <delay> - delay duration in seconds (defaults to 3)
  -s <dbtype> - the database backend to use: goleveldb (default) or couchdb
  -l <language> - the programming language of the chaincode to deploy: go (default), javascript, or java
  -i <imagetag> - the tag to be used to launch the network (defaults to "latest")
  -a - launch certificate authorities (no certificate authorities are launched by default)
  -n - do not deploy chaincode (abstore chaincode is deployed by default)
  -v - verbose mode
  byfn.sh -h (print this message)

Typically, one would first generate the required certificates and
genesis block, then bring up the network. e.g.:

  byfn.sh generate -c mychannel
  byfn.sh up -c mychannel -s couchdb
  byfn.sh up -c mychannel -s couchdb -i 1.4.0
  byfn.sh up -l javascript
  byfn.sh down -c mychannel
  byfn.sh upgrade -c mychannel

Taking all defaults:
  byfn.sh generate
  byfn.sh up
  byfn.sh down

```

Para generar la red **Artifacts** del ejemplo byfn hay que ejecutar el script con la opción generate (./byfn.sh generate), lo cual genera los certificados de las organizaciones (org1 y org2) y el canal de comunicación (mychannel).

```

#####/Fabric/fabric-samples/first-network/./bin/configtxgen
#####
##### Generating Orderer Genesis block #####
#####
2020-03-20 19:14:47.181 CST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-03-20 19:14:47.224 CST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft
2020-03-20 19:14:47.224 CST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRaft.Options unset, set
_inflight_blocks:5 snapshot_interval_size:16777216
2020-03-20 19:14:47.224 CST [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /Fabric/fabric-sa
2020-03-20 19:14:47.228 CST [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2020-03-20 19:14:47.228 CST [common.tools.configtxgen] doOutputBlock -> INFO 006 Writing genesis block

#####
### Generating channel configuration transaction 'channel.tx' ###
#####
+ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID mychannel
2020-03-20 19:14:47.269 CST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-03-20 19:14:47.309 CST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /Fabric/fabric-sa
2020-03-20 19:14:47.309 CST [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 003 Generating new channel configtx
2020-03-20 19:14:47.314 CST [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Writing new channel tx
+ res=0
+ set +x

#####
##### Generating anchor peer update for Org1MSP #####
#####
+ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID mychannel -asOrg Org1MSP
2020-03-20 19:14:47.356 CST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-03-20 19:14:47.382 CST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /Fabric/fabric-sa
2020-03-20 19:14:47.382 CST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-03-20 19:14:47.384 CST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
+ res=0
+ set +x

#####
##### Generating anchor peer update for Org2MSP #####
#####
+ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID mychannel -asOrg Org2MSP
2020-03-20 19:14:47.410 CST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-03-20 19:14:47.432 CST [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /Fabric/fabric-sa
2020-03-20 19:14:47.432 CST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2020-03-20 19:14:47.434 CST [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
+ res=0
+ set +x

```

Para levantar la red se ejecuta la instrucción `./byfn.sh up`