

Conectar SSH

* Instalar Git

sudo apt update

sudo apt upgrade

git --version

sudo apt install git

* Instalar cURL

sudo apt install curl

curl --version

* Instalar Docker

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

sudo docker run hello-world

sudo groupadd Docker

sudo usermod -aG docker $USER

logout

* Volver a conectar con el servidor tras el logout y ejecutar

docker run hello-world

* Docker Compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

* Go (lenguaje de programación)

sudo apt install golang-go

go version

* NPM

sudo apt install npm

Instalación de los componentes de Fabric:

Fabric provee de manera pública un repositorio para poder comenzar a desarrollar y desplegar su tecnología. Este repositorio público será descargado primeramente en una versión más antigua para comprender la evolución que ha sufrido Fabric:

docker pull couchdb

* Descargar couchdb Docker

docker pull couchdb

Descargar repositorios oficiales última versión – Dockers – Binarios o con la versión

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

* Descargar repositorios oficiales (V 2.3.2) – Dockers - Binarios

curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.3.2 1.5.2

* Comprobar que se ha descargado todo correctamente

cd fabric-samples

ls bin

cd

Posteriormente, se procederá a descargar el repositorio preparado para esta formación. Al ser un repositorio público, no será necesario poseer usuario para la descarga. Por ello, habrá que situarse en la carpeta “home” del usuario que se esté usando y descargar el repositorio:

cd

git clone https://github.com/eveporras23/red-blockfarm.git/

cd red-blockfarm

La sintaxis JSON se derivó de JavaScript, pero en sí misma es independiente del lenguaje. Es compatible con muchos lenguajes de programación; estos lenguajes incluyen código que se puede utilizar para incrustar JSON en el programa; pero desafortunadamente, no podemos trabajar con JSON directamente en el shell de Linux ya que no puede interpretarlo. Para trabajar con JSON en el shell de Linux, utilizamos una combinación de herramientas como **JQ** y **sed.**

El comando JQ se utiliza para transformar datos JSON en un formato más legible e imprimirlos en una salida estándar en Linux. El comando JQ se basa en filtros que se utilizan para buscar e imprimir solo los datos requeridos por un archivo JSON.

sudo apt install jq

Posteriormente, se comprobará que están todos los dockers parados y eliminados. A su vez, se eliminarán los volúmenes creados de Docker ya que no se querrán los que se instalaron con anterioridad. De igual manera, se eliminarán todas las redes de Docker que existan para no interceder, si pasase, con las instalaciones posteriores y obtener errores de red con Docker difíciles de detectar.

**docker stop $(docker ps -a -q)**

**docker rm $(docker ps -a -q)**

**docker volume prune**

**docker network prune**

Progresando con la limpieza de la red, habrá que asegurarse que no existen certificados generados con anterioridad y por ello, se eliminarán todas las carpetas con certificados para que tampoco se herede ninguna configuración previa:

sudo rm -rf organizations/peerOrganizations

sudo rm -rf organizations/ordererOrganizations

sudo rm -rf channel-artifacts/

mkdir channel-artifacts

Ahora sí, se podrá comenzar a desplegar.

Una de las carpetas que más importancia adquiere ahora en este despliegue – y en los anteriores ejemplos también – es la carpeta que tiene el nombre “bin”**.** En dicha carpeta se encuentran todos los archivos binarios o ejecutables de los que se servirán los despliegues para poder desplegar los peers, orederers, CA, herramientas para generar la configuración de los canales, etc. Por ello, habrá que exportar la ruta de esta carpeta.

La instrucción posterior, se exportará la configuración inicial de los peers y de los orderers, también será fundamental exportarla.

Con la instalación lista para comenzar y con las variables de entorno necesarias ya exportadas, se procederá a generar los certificados de la red. Para ello se utilizará la herramienta **“cryptogen”** que estas fases iniciales de proyecto o de despliegues sencillos, se puede utilizar. Pero no se recomienda para redes en producción o redes en las que sea necesario realizar una gestión de certificados óptima como se pueda hacer con la CA de Fabric.

Cd red-blockfarm

export PATH=${PWD}/../fabric-samples/bin:${PWD}:$PATH

export FABRIC\_CFG\_PATH=${PWD}/../fabric-samples/config

cryptogen generate --config=./organizations/cryptogen/crypto-config-mapa.yaml --output="organizations"

cryptogen generate --config=./organizations/cryptogen/crypto-config-marketplace.yaml --output="organizations"

cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output="organizations"

Ahora sí, se podrá levantar la red compuesta por nodos virtualizados por los Dockers declarados en la carpeta “docker” y concretamente en el fichero “docker-compose-blockfarm-net.yaml”. Si se desea visualizar detenidamente el fichero, se recomienda navegar a dicha carpeta y acceder al fichero.

docker-compose -f docker/docker-compose-blockfarm-net.yaml up -d

El primer paso para **“presentar”** los nodos y que transaccionen será realizar la **creación de los canales.** Esta creación se realiza con una herramienta que proporciona Hyperledger Fabric para generar ese bloque génesis con la configuración declarada en la carpeta “configtx” y más concretamente en el fichero “configtx.yaml”. Se exportará la carpeta “configtx” para que, al usar la herramienta configtxgen, acceda a dicha carpeta y ejecute las instrucciones posteriores.

export FABRIC\_CFG\_PATH=${PWD}/configtx

configtxgen -profile TwoOrgBlockfarmGenesis -outputCreateChannelTx ./channel-artifacts/twoorgblockfarmchannel.tx -channelID twoorgblockfarmchannel

export FABRIC\_CFG\_PATH=$PWD/../fabric-samples/config/

export CORE\_PEER\_TLS\_ENABLED=true

export CORE\_PEER\_LOCALMSPID=" OrdererBlockFarmMSP"

export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/mapa.blockfarm.com/peers/peer0.mapa.blockfarm.com/tls/ca.crt

export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/mapa.blockfarm.com/users/Admin@mapa.blockfarm.com/msp

export CORE\_PEER\_ADDRESS=localhost:7051

export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/mapa.blockfarm.com/users/Admin@mapa.blockfarm.com/msp

export ORDERER\_CA=${PWD}/organizations/ordererOrganizations/blockfarm.com/orderers/orderer1.blockfarm.com/msp/tlscacerts/tlsca.blockfarm.com-cert.pem

peer channel create -o localhost:7050 --ordererTLSHostnameOverride orderer1.blockfarm.com -c twoorgblockfarmchannel -f ./channel-artifacts/twoorgblockfarmchannel.tx --outputBlock ./channel-artifacts/twoorgblockfarmchannel.block --tls --cafile "$ORDERER\_CA"

export FABRIC\_CFG\_PATH=${PWD}/configtx

configtxgen -profile TwoOrgBlockfarmGenesis -outputBlock ./channel-artifacts/twoorgblockfarmchannel.block -channelID twoorgblockfarmchannel

export FABRIC\_CFG\_PATH=${PWD}/../config

export ORDERER\_CA=${PWD}/organizations/ordererOrganizations/blockfarm.com/orderers/orderer1.blockfarm.com/msp/tlscacerts/tlsca.blockfarm.com-cert.pem

export ORDERER\_ADMIN\_TLS\_SIGN\_CERT=${PWD}/organizations/ordererOrganizations/blockfarm.com/orderers/orderer1.blockfarm.com/tls/server.crt

export ORDERER\_ADMIN\_TLS\_PRIVATE\_KEY=${PWD}/organizations/ordererOrganizations/blockfarm.com/orderers/orderer1.blockfarm.com/tls/server.key

osnadmin channel join --channelID twoorgblockfarmchannel --config-block ./channel-artifacts/twoorgblockfarmchannel.block -o localhost:7050 --ca-file "$ORDERER\_CA" --client-cert "$ORDERER\_ADMIN\_TLS\_SIGN\_CERT" --client-key "$ORDERER\_ADMIN\_TLS\_PRIVATE\_KEY"

osnadmin channel list -o localhost:7050 --ca-file "$ORDERER\_CA" --client-cert "$ORDERER\_ADMIN\_TLS\_SIGN\_CERT" --client-key "$ORDERER\_ADMIN\_TLS\_PRIVATE\_KEY"