Contents

[TỔNG QUAN KIẾN TRÚC HỆ THỐNG 2](#_Toc211495996)

[1. Mạng Blockchain (Hyperledger Fabric Network) 2](#_Toc211495997)

[2. Cơ sở dữ liệu ngân hàng (PostgreSQL Databases) 2](#_Toc211495998)

[3. Hệ thống lưu trữ phi tập trung (IPFS – InterPlanetary File System) 2](#_Toc211495999)

[4. Hệ thống Backend (Bank Server Applications) 3](#_Toc211496000)

[5. Giao diện người dùng (Frontend – React App) 3](#_Toc211496001)

[BỘ CÔNG CỤ ĐỂ CHẠY LAB CHAPTER 4 3](#_Toc211496002)

[LAB 4.0. Chuẩn bị môi trường Hyperledger Fabric 4](#_Toc211496003)

[Kết quả khi chạy thành công 10](#_Toc211496004)

[LAB 4.1. Setting up the hyperledger fabric bankchain network 11](#_Toc211496005)

[Thư mục dự án Bankchain tham khảo: 11](#_Toc211496006)

[1. Tạo crypto-config.yaml file 11](#_Toc211496007)

[file crypto-config.yaml mô tả các tổ chức trong mạng: 11](#_Toc211496008)

[2. Tạo configtx.yaml file 12](#_Toc211496009)

[3. Tạo các docker-compose files: 13](#_Toc211496010)

[4. Launching the network 28](#_Toc211496011)

[LAB 4.2. Creating blockchain identities for the banks 32](#_Toc211496012)

[LAB 4.3. Building the corporate remittance contract 32](#_Toc211496013)

[LAB 4.4. Setting up the IPFS network 32](#_Toc211496014)

[LAB 4.5. Setting up the bank databases 32](#_Toc211496015)

[LAB 4.6. Building the bank backend servers 32](#_Toc211496016)

[LAB 4.7. Building the transaction listeners for the banks 32](#_Toc211496017)

[LAB 4.8. Creating the corporate remittance app frontend in React 32](#_Toc211496018)

# TỔNG QUAN KIẾN TRÚC HỆ THỐNG

Trong Chapter 4 - “**Corporate Remittances and Settlement**”, hệ thống được xây dựng là một mạng blockchain cho chuyển tiền doanh nghiệp (Corporate Remittance Network).   
Hệ thống này gồm 5 nhóm thành phần chính, mỗi nhóm đảm nhận một vai trò cụ thể trong toàn bộ kiến trúc

## 1. Mạng Blockchain (Hyperledger Fabric Network)

- Đây là trung tâm của hệ thống, nơi các giao dịch chuyển tiền được ghi nhận, xác thực và bảo mật.

- Thành phần chính:

* **Orderer Node**: Quản lý việc sắp xếp thứ tự và đóng gói các giao dịch thành block.
* **Peer Nodes (Bank Peers)**: Đại diện cho các ngân hàng trong mạng (ví dụ: Bank A, Bank B). Mỗi ngân hàng có peer riêng để xác thực và lưu trữ bản sao sổ cái.
* **Certificate Authority (CA)**: Cấp chứng chỉ định danh số cho các thành phần (admin, user, client).
* **Channel**: Kênh riêng biệt cho giao dịch giữa các ngân hàng, bảo mật dữ liệu chỉ cho các bên liên quan.
* **Chaincode (Smart Contract)**: Logic nghiệp vụ của chuyển tiền, triển khai trên peer nodes.

 Đây là nơi toàn bộ giao dịch remittance được xử lý và ghi vào blockchain.

## 2. Cơ sở dữ liệu ngân hàng (PostgreSQL Databases)

- Mỗi ngân hàng trong hệ thống có một cơ sở dữ liệu riêng để lưu trữ thông tin khách hàng và giao dịch nội bộ.

- Thành phần chính:

* Customer Table: lưu thông tin khách hàng (tên, ID, tài khoản, địa chỉ,…).
* Transaction Table: ghi lịch sử giao dịch của từng khách hàng.
* Balance Table: theo dõi số dư của từng tài khoản.

Cơ sở dữ liệu này giúp kết hợp dữ liệu on-chain (blockchain) với dữ liệu off-chain (nội bộ ngân hàng) để hiển thị và xử lý linh hoạt.

## 3. Hệ thống lưu trữ phi tập trung (IPFS – InterPlanetary File System)

Dùng để lưu trữ các tài liệu giao dịch, chẳng hạn như hóa đơn, chứng từ tuân thủ (compliance docs), biên lai,…

Thành phần chính:

* **IPFS Nodes:** các node lưu trữ dữ liệu và kết nối thành mạng chia sẻ tệp.
* **Key File & Config File**: cấu hình danh tính và thiết lập mạng IPFS.
* **Utility Scripts**: tiện ích để upload, retrieve file từ IPFS.

IPFS đảm bảo các tệp không thể bị chỉnh sửa, tăng tính minh bạch trong kiểm toán.

## 4. Hệ thống Backend (Bank Server Applications)

- Đây là phần trung gian giữa blockchain, cơ sở dữ liệu, và giao diện người dùng.

- Chức năng chính:

* Giao tiếp với blockchain thông qua SDK của Hyperledger Fabric.
* Gửi và nhận yêu cầu giao dịch.
* Kết nối và lưu dữ liệu vào PostgreSQL.
* Gọi IPFS API để lưu hoặc truy xuất tài liệu.
* Cung cấp RESTful API cho frontend.

- Ví dụ:

* POST /payment-request → tạo yêu cầu chuyển tiền mới.
* GET /transactions → truy xuất danh sách giao dịch.
* PUT /balance → cập nhật số dư khách hàng.

 Backend giúp kết nối toàn bộ các lớp trong hệ thống lại với nhau.

## 5. Giao diện người dùng (Frontend – React App)

- Đây là ứng dụng web dành cho nhân viên ngân hàng và người dùng doanh nghiệp để tương tác với hệ thống.

- Thành phần chính:

* **Login Component**: xác thực người dùng (Bank Admin hoặc Customer).
* **Transfer Component**: tạo yêu cầu chuyển tiền.
* **View Transactions Component**: xem lịch sử và trạng thái giao dịch.
* **Compliance Upload**: tải lên và xem tài liệu từ IPFS.

Giao diện này giao tiếp với backend qua API, hiển thị dữ liệu on-chain và off-chain một cách trực quan.

**Tóm lại, hệ thống trong Chapter 4 gồm:**

* **Blockchain Layer Hyperledger Fabric** (Orderer, Peers, CA, Channel, Chaincode): Xử lý và ghi nhận giao dịch trên sổ cái phân tán.
* **Database Layer** (PostgreSQL): Lưu trữ dữ liệu khách hàng & số dư ngoài chuỗi.
* **Storage Layer IPFS**: Lưu trữ tài liệu giao dịch phi tập trung.
* **Backend Layer** (Node.js/Express APIs): Trung gian kết nối blockchain – database - frontend.
* **Frontend Layer** (ReactJS App): Giao diện người dùng cho ngân hàng & khách hàng.

# BỘ CÔNG CỤ ĐỂ CHẠY LAB CHAPTER 4

**1. Hyperledger Fabric Core**

* **Peer / Orderer / Tools (cryptogen, configtxgen, peer, orderer)** Version: **2.2.x (LTS)** (ví dụ 2.2.0 hoặc 2.2.5)
* Docker images:
  + hyperledger/fabric-peer:2.2
  + hyperledger/fabric-orderer:2.2
  + hyperledger/fabric-tools:2.2

**2. Hyperledger Fabric CA**

* Vẫn dùng bản cũ **1.4.x**, vì Fabric-CA chưa lên v2.
  + Docker image: hyperledger/fabric-ca:1.4.9

**3. Database cho peer state**

* **CouchDB**: 3.1.1 (theo docker-compose trong sách) Image: couchdb:3.1.1

**4. Ngôn ngữ chaincode**

* **Go** (dùng để viết chaincode backend) Version: **1.14.x** hoặc **1.15.x** (không dùng bản quá mới vì có thể lỗi với Fabric 2.2)
* Ngoài Go, chaincode cũng có thể viết bằng Node.js, nhưng sách chủ yếu demo bằng Go.

**5. Node.js (cho backend & frontend)**

* **Node.js v12.x LTS** Đúng theo hướng dẫn trong sách (curl -sL https://deb.nodesource.com/setup\_12.x)
* **npm** đi kèm với Node.js

**6. Docker & Docker Compose**

* **Docker CE** mới nhất (20.x trở lên).
* **Docker Compose v1.27.x** hoặc v1.29.x (chưa cần bản 2.x).

**7. Các công cụ khác**

* **cURL** (để test REST API).
* **Git** (clone fabric-samples, project code).

# LAB 4.0. Chuẩn bị môi trường Hyperledger Fabric

**1. Cài Máy ảo Ubuntu:**

**Ubuntu 24.04.3 LTS:**

<https://ubuntu.com/download/desktop/thank-you?version=24.04.3&architecture=amd64&lts=true>

**2. Chuẩn bị môi trường (Ubuntu)**

**-** Cập nhật hệ thống

|  |
| --- |
| sudo apt update |

* **Mục đích:** Cập nhật lại danh sách package (gói phần mềm) từ các kho (repository) mà Ubuntu đang cấu hình.
* **Ý nghĩa:** Không cài gì cả, chỉ tải mới thông tin "gói nào có phiên bản gì" về máy.
* **Ví dụ:** Nếu có bản mới của curl hay docker.io, lệnh này sẽ giúp hệ thống biết để cài bản mới nhất.

**-** Cài đặt các công cụ cần thiết: curl, git, docker.io:

|  |
| --- |
| sudo apt install -y curl  sudo apt install -y git  sudo apt install -y docker.io |

sudo apt install -y curl

* **Mục đích:** Cài đặt công cụ curl.
* **curl:** một tool dòng lệnh để tải dữ liệu từ Internet (HTTP, HTTPS, FTP, …). Trong lab bạn cần nó để tải script, file cấu hình, và binary từ GitHub/Hyperledger.
* **Tùy chọn -y:** Tự động trả lời “Yes” cho các câu hỏi xác nhận khi cài đặt.

sudo apt install -y git

* **Mục đích:** Cài đặt git.
* **git:** công cụ quản lý mã nguồn (version control system), dùng để clone repo fabric-samples từ GitHub về máy để thực hành.

sudo apt install -y docker.io

* **Mục đích:** Cài đặt docker.io.
* **docker.io:** gói cài Docker từ kho chính thức của Ubuntu. Docker dùng để chạy các container (peer, orderer, CA, CouchDB trong Hyperledger Fabric).
* Sau khi cài, bạn có thể kiểm tra bằng: docker --version

**Tóm lại:**

* apt update → làm mới danh sách gói.
* curl → tải file từ Internet.
* git → clone code từ GitHub.
* docker.io → chạy container (cốt lõi của Fabric network).

Kiểm tra:

|  |
| --- |
| curl --version  git --version  docker --version |

- Cài **docker-compose:**

* Tải file Compose" V2 **mới nhất** cho x86\_64 và đặt vào folder plugin:

|  |
| --- |
| sudo mkdir -p /usr/local/lib/docker/cli-plugins  sudo curl -fSL "https://github.com/docker/compose/releases/latest/download/docker-compose-linux-x86\_64" \  -o /usr/local/lib/docker/cli-plugins/docker-compose  sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose |

* + Tạo thư mục chứa plugin: sudo mkdir -p /usr/local/lib/docker/cli-plugins
* sudo: chạy với quyền quản trị.
* mkdir -p: tạo thư mục; có -p thì nếu cha chưa có sẽ tạo luôn, còn nếu đã tồn tại thì không báo lỗi.
* /usr/local/lib/docker/cli-plugins: đường dẫn mà Docker tìm plugin của CLI. Đặt Compose ở đây thì lệnh docker compose sẽ hoạt động.
  + Tải Docker Compose (binary) về đúng thư mục plugin: sudo curl -fSL "https://github.com/docker/compose/releases/latest/download/docker-compose-linux-x86\_64" -o /usr/local/lib/docker/cli-plugins/docker-compose
* curl: tải file từ internet.
* -f: “fail on error” → nếu HTTP lỗi (>=400) thì thoát với mã lỗi.
* -S: hiển thị thông báo lỗi chi tiết khi thất bại.
* -L: theo dõi redirect (GitHub release dùng redirect).
* URL "…/latest/download/docker-compose-linux-x86\_64": luôn trỏ đến bản mới nhất cho kiến trúc x86\_64 (amd64).
* -o /usr/local/lib/docker/cli-plugins/docker-compose: ghi file tải về vào đúng vị trí và tên “docker-compose”.
  + **Bật quyền thực thi** cho file docker-compose vừa tải về, để hệ thống coi nó là chương trình có thể chạy: sudo chmod +x /usr/local/lib/docker/cli-plugins/docker-compose
    - sudo: cần quyền quản trị vì file nằm trong /usr/local/lib/....
    - chmod +x: thêm ( + ) quyền execute (x) cho file (thường cho owner, group, others theo umask).
    - Kết quả: lệnh docker compose mới chạy được.
* Kiểm tra:

|  |
| --- |
| docker --version |

* Thêm user của bạn vào nhóm docker để chạy docker không cần sudo:

|  |
| --- |
| sudo usermod -aG docker $USER |

- Cài cài Node.js 22.x LTS trên Ubuntu

* + Thêm NodeSource repo cho Node.js 22.x:

|  |
| --- |
| curl -fsSL https://deb.nodesource.com/setup\_22.x | sudo -E bash - |

Lệnh này sẽ:

* Thêm repository NodeSource vào hệ thống.
* Import GPG key cần thiết.
  + Cài Node.js:

|  |
| --- |
| sudo apt-get install nodejs -y |

=> Gói nodejs sẽ bao gồm cả npm.

* + Kiểm tra phiên bản:

|  |
| --- |
| nodejs -v  npm -v |

**- Cài Go (Golang)** trên Ubuntu 18.04/20.04

* Trong **Chapter 4**, chaincode được viết bằng **Go** → nên bắt buộc phải cài Go.
* Ngoài ra, nhiều công cụ CLI của Fabric (như peer) cũng build dựa trên Go.
* **Hyperledger Fabric 3.1.x** tương thích nhất với: **Go 1.24.0**
* Tải Go 1.24

|  |
| --- |
| wget https://go.dev/dl/go1.24.0.linux-amd64.tar.gz |

* Giải nén vào /usr/local

|  |
| --- |
| sudo tar -C /usr/local -xzf go1.24.0.linux-amd64.tar.gz |

* Thêm Go vào PATH

|  |
| --- |
| export GOROOT=/usr/local/go  export GOPATH=$HOME/go  export PATH=$PATH:$GOROOT/bin:$GOPATH/bin |

* Áp dụng thay đổi:

|  |
| --- |
| source ~/.profile |

* Kiểm tra phiên bản đã cài:

|  |
| --- |
| go version |

**- Cài Hyperledger Fabric 3.1.x**

* Tải Fabric samples mới nhất:

|  |
| --- |
| git clone https://github.com/hyperledger/fabric-samples.git  cd fabric-samples |

* Chạy script cài binaries và Docker images cho Fabric 3.1.x:

|  |
| --- |
| curl -sSL https://bit.ly/2ysbOFE | bash -s -- 3.1.0 1.5.10 |

* https://bit.ly/2ysbOFE → redirect đến scripts/bootstrap.sh trong repo Hyperledger.
* 3.1.0 → phiên bản **Fabric binaries + Docker images**.
* 1.5.10 → phiên bản **Fabric CA** tương ứng.

Sau khi cài xong, bạn sẽ có Thư mục:

fabric-samples/

├── bin/

├── config/

└── ...

Các binary chính: peer, orderer, configtxgen, configtxlator, cryptogen, discover, osnadmin, fabric-ca-client

Các Docker images tương ứng (Fabric 3.1.x):

hyperledger/fabric-peer:3.1

hyperledger/fabric-orderer:3.1

hyperledger/fabric-tools:3.1

hyperledger/fabric-ccenv:3.1

hyperledger/fabric-ca:1.5.10

* Kiểm tra:

|  |
| --- |
| docker images | grep hyperledger |

Nếu đánh lệnh trên không thấy hiển thị gì thì nghĩa là Script bootstrap.sh chỉ tải phần **binaries**, nhưng **chưa tải được Docker images** (có thể do mạng, Docker Hub rate-limit, hoặc script bị ngắt giữa chừng).

Cách khắc phục: Chạy lại phần tải image **riêng** (an toàn, không ảnh hưởng phần bin/): Kéo thủ công các images cần thiết cho Fabric 3.1.x

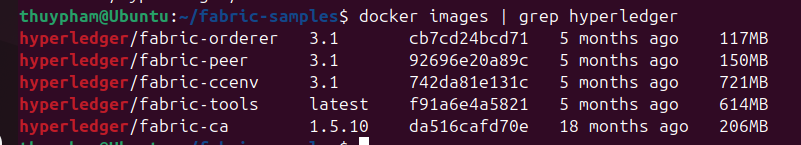
|  |
| --- |
| docker pull hyperledger/fabric-peer:3.1  docker pull hyperledger/fabric-orderer:3.1  docker pull hyperledger/fabric-tools:3.1  docker pull hyperledger/fabric-ccenv:3.1  docker pull hyperledger/fabric-ca:1.5.10 |

Sau khi kéo xong kiểm tra lại:

|  |
| --- |
| docker images | grep hyperledger |

Kết quả mong đợi:

|  |
| --- |
| hyperledger/fabric-peer 3.1  hyperledger/fabric-orderer 3.1  hyperledger/fabric-tools 3.1  hyperledger/fabric-ccenv 3.1  hyperledger/fabric-ca 1.5.10 |



Nếu thiếu cái nào trong list trên thì kéo tiếp: Ví dụ thiếu hyperledger/fabric-tools 3.1:

|  |
| --- |
| docker pull hyperledger/fabric-tools:latest |

* Thêm binaries vào PATH;

|  |
| --- |
| export PATH=${PWD}/bin:$PATH |

Để dùng vĩnh viễn:

|  |
| --- |
| echo 'export PATH=${PWD}/bin:$PATH' >> ~/.bashrc  source ~/.bashrc |

* Kiểm tra kết quả:

|  |
| --- |
| peer version  orderer version  fabric-ca-client version |

Nếu đánh lệnhpeer version báo lỗi “peer not found”…là do thiếu bước thêm đường dẫn vào PATH, nên Ubuntu vẫn chưa “nhìn thấy” lệnh peer. Thêm đường dẫn bằng lệnh:

|  |
| --- |
| export PATH=$HOME/fabric-samples/bin:$PATH |

Sau đó thử lại:

|  |
| --- |
| peer version |

Nếu lệnh này chạy được (in ra version của Fabric Peer), nghĩa là OK.

Để lệnh peer luôn dùng được (khỏi export lại mỗi lần mở terminal), Thêm dòng export này vào cuối file ~/.bashrc

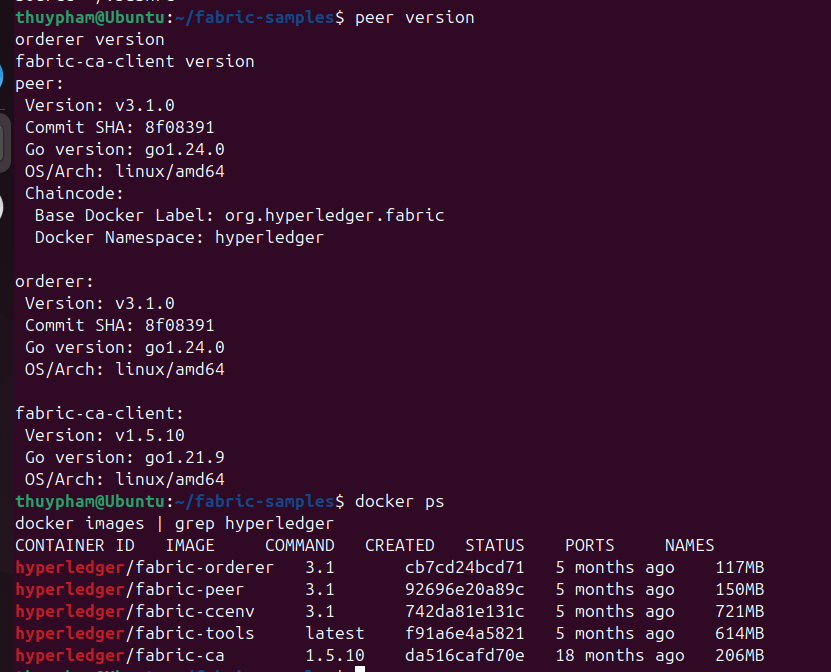
|  |
| --- |
| echo 'export PATH=$HOME/fabric-samples/bin:$PATH' >> ~/.bashrc  source ~/.bashrc |

Kiểm tra thêm:

|  |
| --- |
| which peer |

Kết quả mong đợi: /home/<username>/fabric-samples/bin/peer

## **Kết quả khi chạy thành công**



Đây là bộ toolset được sử dụng trong Chapter 4 để dựng mạng **Bankchain**.

# LAB 4.1. ****Setting up the hyperledger fabric bankchain network****

**⬄** dựng một **mạng Fabric** (gồm orderer, peer, CA, channel…) để làm nền cho các bước sau.

**Thư mục dự án Bankchain tham khảo:**

Tải code mẫu từ GitHub:

|  |
| --- |
| git clone https://github.com/PacktPublishing/Blockchain-Development-for-Finance-Projects.git  cd Blockchain-Development-for-Finance-Projects/Chapter4/bankchain |

Có thể **sao chép thư mục** bankchain vào trong fabric-samples (để có sẵn binaries và config)

## 1. Tạo crypto-config.yaml file

### file crypto-config.yaml mô tả các tổ chức trong mạng:

* **OrdererOrg** (có 1 orderer).
* **Bank1, Bank2** (mỗi org có 2 peer và 1 user admin).

Nó giống như “bản sơ đồ nhân sự” của mạng blockchain

crypto-config.yaml:

|  |
| --- |
| OrdererOrgs:  - Name: Orderer  Domain: bankchain.com  EnableNodeOUs: true  Specs:  - Hostname: orderer  PeerOrgs:  - Name: BankA  Domain: banka.bankchain.com  EnableNodeOUs: true  Template:  Count: 2  Users:  Count: 1  - Name: BankB  Domain: bankb.bankchain.com  EnableNodeOUs: true  Template:  Count: 2  Users:  Count: 1 |

## 2. Tạo configtx.yaml file

**File được tạo ở folder** bankchain. File này:

* Định nghĩa **consortium** (BankConsortium gồm Bank1 + Bank2).
* Định nghĩa **anchor peer** cho từng ngân hàng.
* Đây chính là “luật chơi” của mạng blockchain: ai tham gia, quyền gì, kết nối thế nào.

configtx.yaml:

|  |
| --- |
| Organizations:  - &OrdererOrg  Name: OrdererMSP  ID: OrdererMSP  MSPDir: crypto-config/ordererOrganizations/bankchain.com/msp  - &BankA  Name: BankAMSP  ID: BankAMSP  MSPDir: crypto-config/peerOrganizations/banka.bankchain.com/msp  AnchorPeers:  - Host: peer0.banka.bankchain.com  Port: 7051  - &BankB  Name: BankBMSP  ID: BankBMSP  MSPDir: crypto-config/peerOrganizations/bankb.bankchain.com/msp  AnchorPeers:  - Host: peer0.bankb.bankchain.com  Port: 8051  Orderer:  OrdererType: solo  Addresses:  - orderer.bankchain.com:7050  Organizations:  - \*OrdererOrg  Application:  Organizations:  Profiles:  BankchainGenesis:  Orderer:  OrdererType: solo  Addresses:  - orderer.bankchain.com:7050  Organizations:  - \*OrdererOrg  Consortiums:  BankConsortium:  Organizations:  - \*BankA  - \*BankB  BankChannel:  Consortium: BankConsortium  Application:  Organizations:  - \*BankA  - \*BankB |

## 3. Tạo các docker-compose files:

* docker-compose-bankchain.yaml: Defines the parameters used to bring the orderer and the peer containers online.

The containers are orderer.example.com, peer0.banka.example.com, peer1.banka.example.com, peer0.bankb.example.com, and peer1.bankb.example.com.

Additionally, it also creates the cli container, which hosts command-line tools, used to interact with the peer0.banka.example.com container.

* docker-compose-ca.yaml: Defines the parameters used to bring the **certificate authorities** (**CA**) online. We'll be running two CAs, ca0 and ca1, one for each organization, Bank A and Bank B. The containers are named ca\_peerbanka and ca\_peerbankb.
* docker-compose-couch.yaml: Defines the parameters used to bring the CouchDB database online. The couch database stores the transaction data and the state database for a peer node. We'll be running four databases, one for each peer—couchdb0, couchdb1, couchdb2, and couchdb3.

docker-compose-bankchain.yaml:

|  |
| --- |
| version: '2'  volumes:  orderer.example.com:  peer0.banka.example.com:  peer1.banka.example.com:  peer0.bankb.example.com:  peer1.bankb.example.com:  networks:  bankchain:  services:  orderer.example.com:  extends:  file: base/docker-compose-base-bankchain.yaml  service: orderer.example.com  container\_name: orderer.example.com  networks:  - bankchain  peer0.banka.example.com:  container\_name: peer0.banka.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer0.banka.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb0:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb0  peer1.banka.example.com:  container\_name: peer1.banka.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer1.banka.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb1:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb1  peer0.bankb.example.com:  container\_name: peer0.bankb.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer0.bankb.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb2:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb2  peer1.bankb.example.com:  container\_name: peer1.bankb.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer1.bankb.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb3:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb3    cli:  container\_name: cli  image: hyperledger/fabric-tools:$IMAGE\_TAG  tty: true  stdin\_open: true  environment:  - SYS\_CHANNEL=$SYS\_CHANNEL  - GOPATH=/opt/gopath  - CORE\_VM\_ENDPOINT=unix:///host/var/run/docker.sock  #- FABRIC\_LOGGING\_SPEC=DEBUG  - FABRIC\_LOGGING\_SPEC=INFO  - CORE\_PEER\_ID=cli  - CORE\_PEER\_ADDRESS=peer0.banka.example.com:7051  - CORE\_PEER\_LOCALMSPID=BankAMSP  - CORE\_PEER\_TLS\_ENABLED=true  - CORE\_PEER\_TLS\_CERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.example.com/peers/peer0.banka.example.com/tls/server.crt  - CORE\_PEER\_TLS\_KEY\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.example.com/peers/peer0.banka.example.com/tls/server.key  - CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.example.com/peers/peer0.banka.example.com/tls/ca.crt  - CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.example.com/users/Admin@banka.example.com/msp  working\_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer  command: /bin/bash  volumes:  - /var/run/:/host/var/run/  - ./../chaincode/:/opt/gopath/src/github.com/chaincode  - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/  - ./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/  - ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts  depends\_on:  - orderer.example.com  - peer0.banka.example.com  - peer1.banka.example.com  - peer0.bankb.example.com  - peer1.bankb.example.com  networks:  - bankchain  couchdb0:  container\_name: couchdb0  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "5984:5984"  networks:  - bankchain      couchdb1:  container\_name: couchdb1  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "6984:5984"  networks:  - bankchain    couchdb2:  container\_name: couchdb2  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "7984:5984"  networks:  - bankchain    couchdb3:  container\_name: couchdb3  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "8984:5984"  networks:  - bankchain |

docker-compose-ca.yaml:

|  |
| --- |
| version: '2'  volumes:  orderer.example.com:  peer0.banka.example.com:  peer1.banka.example.com:  peer0.bankb.example.com:  peer1.bankb.example.com:  networks:  bankchain:  services:  orderer.example.com:  extends:  file: base/docker-compose-base-bankchain.yaml  service: orderer.example.com  container\_name: orderer.example.com  networks:  - bankchain  peer0.banka.example.com:  container\_name: peer0.banka.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer0.banka.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb0:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb0  peer1.banka.example.com:  container\_name: peer1.banka.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer1.banka.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb1:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb1  peer0.bankb.example.com:  container\_name: peer0.bankb.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer0.bankb.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb2:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb2  peer1.bankb.example.com:  container\_name: peer1.bankb.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer1.bankb.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb3:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb3      ca0:  image: hyperledger/fabric-ca:$IMAGE\_TAG  environment:  - FABRIC\_CA\_HOME=/etc/hyperledger/fabric-ca-server  - FABRIC\_CA\_SERVER\_CA\_NAME=ca-banka  - FABRIC\_CA\_SERVER\_TLS\_ENABLED=true  - FABRIC\_CA\_SERVER\_TLS\_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.banka.example.com-cert.pem  - FABRIC\_CA\_SERVER\_TLS\_KEYFILE=/etc/hyperledger/fabric-ca-server-config/${BANKCHAIN\_CA1\_PRIVATE\_KEY}  - FABRIC\_CA\_SERVER\_PORT=7054  ports:  - "7054:7054"  command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.banka.example.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/${BANKCHAIN\_CA1\_PRIVATE\_KEY} -b admin:adminpw -d'  volumes:  - ./crypto-config/peerOrganizations/banka.example.com/ca/:/etc/hyperledger/fabric-ca-server-config  container\_name: ca\_peerbanka  networks:  - bankchain  ca1:  image: hyperledger/fabric-ca:$IMAGE\_TAG  environment:  - FABRIC\_CA\_HOME=/etc/hyperledger/fabric-ca-server  - FABRIC\_CA\_SERVER\_CA\_NAME=ca-bankb  - FABRIC\_CA\_SERVER\_TLS\_ENABLED=true  - FABRIC\_CA\_SERVER\_TLS\_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.bankb.example.com-cert.pem  - FABRIC\_CA\_SERVER\_TLS\_KEYFILE=/etc/hyperledger/fabric-ca-server-config/${BANKCHAIN\_CA2\_PRIVATE\_KEY}  - FABRIC\_CA\_SERVER\_PORT=8054  ports:  - "8054:8054"  command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.bankb.example.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/${BANKCHAIN\_CA2\_PRIVATE\_KEY} -b admin:adminpw -d'  volumes:  - ./crypto-config/peerOrganizations/bankb.example.com/ca/:/etc/hyperledger/fabric-ca-server-config  container\_name: ca\_peerbankb  networks:  - bankchain  couchdb0:  container\_name: couchdb0  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "5984:5984"  networks:  - bankchain    couchdb1:  container\_name: couchdb1  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "6984:5984"  networks:  - bankchain    couchdb2:  container\_name: couchdb2  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "7984:5984"  networks:  - bankchain    couchdb3:  container\_name: couchdb3  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "8984:5984"  networks:  - bankchain |

docker-compose-couch.yaml:

|  |
| --- |
| version: '2'  networks:  bankchain:  services:    couchdb0:  container\_name: couchdb0  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "5984:5984"  networks:  - bankchain  peer0.banka.example.com:  container\_name: peer0.banka.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer0.banka.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb0:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb0  couchdb1:  container\_name: couchdb1  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "6984:5984"  networks:  - bankchain  peer1.banka.example.com:  container\_name: peer1.banka.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer1.banka.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb1:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb1      couchdb2:  container\_name: couchdb2  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "7984:5984"  networks:  - bankchain  peer0.bankb.example.com:  container\_name: peer0.bankb.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer0.bankb.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb2:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb2  couchdb3:  container\_name: couchdb3  image: hyperledger/fabric-couchdb  # Populate the COUCHDB\_USER and COUCHDB\_PASSWORD to set an admin user and password  # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.  environment:  - COUCHDB\_USER=  - COUCHDB\_PASSWORD=  # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,  # for example map it to utilize Fauxton User Interface in dev environments.  ports:  - "8984:5984"  networks:  - bankchain    peer1.bankb.example.com:  container\_name: peer1.bankb.example.com  extends:  file: base/docker-compose-base-bankchain.yaml  service: peer1.bankb.example.com  networks:  - bankchain  environment:  - CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=couchdb3:5984  # The CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME and CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD  # provide the credentials for ledger to connect to CouchDB. The username and password must  # match the username and password set for the associated CouchDB.  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=  - CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=  depends\_on:  - couchdb3 |

## 4. Launching the network

Thực hiện 4 bước sau:

* **Generate certificates** cho mọi participant bằng cryptogen.
* **Generate network artifacts** (gồm genesis block) bằng configtxgen.
* **docker-compose up** với toàn bộ YAML liên quan để khởi chạy container.
* **Tạo channel + E2E test** từ container cli để chắc mạng/peer đã chạy.

Để thực hiện 4 bước trên ta có thể làm 2 cách:

**Cách 1: dùng script có sẵn:**

|  |
| --- |
| cd ~/fabric-samples/bankchain  ./startBankchain.sh  docker ps # mong đợi ~11 container đang chạy |

**Cách 2: làm thủ công từng bước**

Giả sử bạn đang ở: cd ~/fabric-samples/bankchain   
(Nếu cryptogen, configtxgen nằm trong ~/fabric-samples/bin thì thêm PATH: export PATH=~/fabric-samples/bin:$PATH)

**Chuẩn bị:**

|  |
| --- |
| # Đứng ở thư mục bankchain (nơi có crypto-config.yaml, configtx.yaml, các compose \*.yaml)  export FABRIC\_CFG\_PATH=$PWD  # Thư mục output  rm -rf crypto-config channel-artifacts && mkdir -p crypto-config channel-artifacts  # Tag image bạn đang dùng  export IMAGE\_TAG=3.1.0  # (nếu có chạy CA thì) tag CA ổn định  export CA\_IMAGE\_TAG=1.5.7  # Kéo trước image để tránh “manifest unknown”  docker pull ghcr.io/hyperledger/fabric-peer:$IMAGE\_TAG  docker pull ghcr.io/hyperledger/fabric-orderer:$IMAGE\_TAG  docker pull couchdb:3.3  # Nếu dùng Fabric-CA:  docker pull hyperledger/fabric-ca:$CA\_IMAGE\_TAG |

**Bước 1**: Generate certificates (cryptogen)

|  |
| --- |
| # Sinh toàn bộ MSP/cert theo crypto-config.yaml  cryptogen generate --config=./crypto-config.yaml --output=./crypto-config |

**Bước 2:** Generate channel artifacts (configtxgen – v3 không có system-channel)

Ở Fabric v3, orderer sẽ **join** vào application channel đầu tiên bằng Participation API (osnadmin).  
Ta tạo **genesis block của channel** luôn:

|  |
| --- |
| # Tạo genesis block cho kênh bkchannel (đặt đúng profile trong configtx.yaml)  configtxgen -configPath $PWD \  -profile TwoOrgsChannel \  -channelID bkchannel \  -outputBlock ./channel-artifacts/bkchannel.block  # (khuyến nghị) Tạo bản cập nhật anchor peer cho từng org  configtxgen -configPath $PWD -profile TwoOrgsChannel -channelID bkchannel \  -asOrg BankAMSP -outputAnchorPeersUpdate ./channel-artifacts/BankAMSPanchors.tx  configtxgen -configPath $PWD -profile TwoOrgsChannel -channelID bkchannel \  -asOrg BankBMSP -outputAnchorPeersUpdate ./channel-artifacts/BankBMSPanchors.tx |

**Bước 3:** Khởi động network (docker compose up):

Ba file compose được dùng: docker-compose-bankchain.yaml, docker-compose-ca.yaml, docker-compose-couch.yaml

|  |
| --- |
| # kéo image trước  docker compose \  -f docker-compose-bankchain.yaml \  -f docker-compose-ca.yaml \  -f docker-compose-couch.yaml \  pull  # khởi động tất cả (orderer + peers + CA + CouchDB)  docker compose \  -f docker-compose-bankchain.yaml \  -f docker-compose-ca.yaml \  -f docker-compose-couch.yaml \  up -d |

Nếu cần dừng/xóa sạch thì làm như sau:

|  |
| --- |
| docker compose \  -f docker-compose-bankchain.yaml \  -f docker-compose-ca.yaml \  -f docker-compose-couch.yaml \  down -v |

*Lưu ý cho Fabric v3:* trong **orderer** service phải có:

* ORDERER\_GENERAL\_BOOTSTRAPMETHOD=none
* ORDERER\_CHANNELPARTICIPATION\_ENABLED=true
* ORDERER\_ADMIN\_LISTENADDRESS=0.0.0.0:7053 (cổng admin)

**Bước 4:** Tạo channel + E2E test từ container **cli**

4.1. Orderer **join** vào channel (Participation API – osnadmin):

|  |
| --- |
| # copy genesis block vào trong container orderer (đường dẫn bên dưới có thể khác nhẹ tùy compose)  docker cp ./channel-artifacts/bkchannel.block \  orderer.bankchain.com:/var/hyperledger/orderer/bkchannel.block  # chạy osnadmin \*bên trong\* orderer container để join  docker exec -it orderer.bankchain.com bash -lc \  'osnadmin channel join \  --channelID bkchannel \  --config-block /var/hyperledger/orderer/bkchannel.block \  -o localhost:7053 \  --ca-file /var/hyperledger/orderer/tls/ca.crt \  --client-cert /var/hyperledger/orderer/tls/server.crt \  --client-key /var/hyperledger/orderer/tls/server.key'  # kiểm tra orderer đã có channel  docker exec orderer.bankchain.com osnadmin channel list -o localhost:7053 \  --ca-file /var/hyperledger/orderer/tls/ca.crt \  --client-cert /var/hyperledger/orderer/tls/server.crt \  --client-key /var/hyperledger/orderer/tls/server.key |

4.2. Peers join channel (từ container cli):

|  |
| --- |
| # vào cli  docker exec -it cli bash  # --- trong CLI ---  export CHANNEL\_NAME=bkchannel  export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/bankchain.com/orderers/orderer.bankchain.com/tls/ca.crt  # Bank A - peer0 (CLI đã trỏ sẵn đến peer0.banka… theo docker-compose)  peer channel join -b ./channel-artifacts/bkchannel.block --tls --cafile $ORDERER\_CA  # Bank A - peer1  export CORE\_PEER\_LOCALMSPID=BankAMSP  export CORE\_PEER\_ADDRESS=peer1.banka.bankchain.com:8051  export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/users/Admin@banka.bankchain.com/msp  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/peers/peer1.banka.bankchain.com/tls/ca.crt  peer channel join -b ./channel-artifacts/bkchannel.block --tls --cafile $ORDERER\_CA  # Bank B - peer0  export CORE\_PEER\_LOCALMSPID=BankBMSP  export CORE\_PEER\_ADDRESS=peer0.bankb.bankchain.com:9051  export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/users/Admin@bankb.bankchain.com/msp  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/peers/peer0.bankb.bankchain.com/tls/ca.crt  peer channel join -b ./channel-artifacts/bkchannel.block --tls --cafile $ORDERER\_CA  # Bank B - peer1  export CORE\_PEER\_ADDRESS=peer1.bankb.bankchain.com:10051  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/peers/peer1.bankb.bankchain.com/tls/ca.crt  peer channel join -b ./channel-artifacts/bkchannel.block --tls --cafile $ORDERER\_CA |

4.3. Cập nhật anchor peers (mỗi org một lần):

|  |
| --- |
| # Bank A  export CORE\_PEER\_LOCALMSPID=BankAMSP  export CORE\_PEER\_ADDRESS=peer0.banka.bankchain.com:7051  export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/users/Admin@banka.bankchain.com/msp  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/peers/peer0.banka.bankchain.com/tls/ca.crt  peer channel update -o orderer.bankchain.com:7050 -c bkchannel \  -f ./channel-artifacts/BankAMSPanchors.tx --tls --cafile $ORDERER\_CA  # Bank B  export CORE\_PEER\_LOCALMSPID=BankBMSP  export CORE\_PEER\_ADDRESS=peer0.bankb.bankchain.com:9051  export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/users/Admin@bankb.bankchain.com/msp  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/peers/peer0.bankb.bankchain.com/tls/ca.crt  peer channel update -o orderer.bankchain.com:7050 -c bkchannel \  -f ./channel-artifacts/BankBMSPanchors.tx --tls --cafile $ORDERER\_CA |

4.4 E2E test (deploy “asset-transfer-basic” – Go):

Thư mục chaincode trong compose đã map vào /opt/gopath/src/github.com/chaincode.  
Điều chỉnh --path nếu bạn để chaincode ở vị trí khác.

|  |
| --- |
| # --- vẫn bên trong CLI ---  # 1) Package  peer lifecycle chaincode package basic.tar.gz \  --lang golang \  --label basic\_1 \  --path /opt/gopath/src/github.com/chaincode/asset-transfer-basic/chaincode-go  # 2) Cài đặt lên từng peer  # Bank A peer0  export CORE\_PEER\_LOCALMSPID=BankAMSP  export CORE\_PEER\_ADDRESS=peer0.banka.bankchain.com:7051  export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/users/Admin@banka.bankchain.com/msp  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/peers/peer0.banka.bankchain.com/tls/ca.crt  peer lifecycle chaincode install basic.tar.gz  # Bank B peer0  export CORE\_PEER\_LOCALMSPID=BankBMSP  export CORE\_PEER\_ADDRESS=peer0.bankb.bankchain.com:9051  export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/users/Admin@bankb.bankchain.com/msp  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/peers/peer0.bankb.bankchain.com/tls/ca.crt  peer lifecycle chaincode install basic.tar.gz  # 3) Lấy PACKAGE\_ID  peer lifecycle chaincode queryinstalled | grep "basic\_1:" | sed -n 's/Package ID: \(.\*\), Label: .\*/export PKGID=\1/p'  echo $PKGID  # 4) Approve cho từng org  # Bank A  export CORE\_PEER\_LOCALMSPID=BankAMSP  export CORE\_PEER\_ADDRESS=peer0.banka.bankchain.com:7051  peer lifecycle chaincode approveformyorg -o orderer.bankchain.com:7050 \  --channelID bkchannel --name basic --version 1.0 --sequence 1 \  --package-id $PKGID --tls --cafile $ORDERER\_CA  # Bank B  export CORE\_PEER\_LOCALMSPID=BankBMSP  export CORE\_PEER\_ADDRESS=peer0.bankb.bankchain.com:9051  peer lifecycle chaincode approveformyorg -o orderer.bankchain.com:7050 \  --channelID bkchannel --name basic --version 1.0 --sequence 1 \  --package-id $PKGID --tls --cafile $ORDERER\_CA  # 5) Commit (kèm địa chỉ & TLS root của mỗi peer tổ chức để xác nhận)  peer lifecycle chaincode commit -o orderer.bankchain.com:7050 \  --channelID bkchannel --name basic --version 1.0 --sequence 1 \  --tls --cafile $ORDERER\_CA \  --peerAddresses peer0.banka.bankchain.com:7051 \  --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/banka.bankchain.com/peers/peer0.banka.bankchain.com/tls/ca.crt \  --peerAddresses peer0.bankb.bankchain.com:9051 \  --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/bankb.bankchain.com/peers/peer0.bankb.bankchain.com/tls/ca.crt  # 6) Invoke & Query nhanh  peer chaincode invoke -o orderer.bankchain.com:7050 --tls --cafile $ORDERER\_CA \  -C bkchannel -n basic -c '{"Args":["InitLedger"]}'  sleep 3  peer chaincode query -C bkchannel -n basic -c '{"Args":["GetAllAssets"]}' |

### Lưu ý khi chạy:

* Port peers có thể khác (7051/8051/9051/10051). Dùng đúng port bạn cấu hình trong base/docker-compose-base-bankchain.yaml.
* Mọi đường dẫn TLS/MSP trong lệnh đều theo domain **bankchain.com** như đã đặt.
* Bố cục 4 bước giống như hướng dẫn trong sách, nhưng phần tạo channel đã đổi sang **Participation API (osnadmin)** vì Fabric v3 bỏ system-channel.

**Kết quả LAB 4.1:**

* Mạng Bankchain chạy với **1 Orderer + 2 Org (Bank1, Bank2)**.
* Có channel bankchannel kết nối 2 ngân hàng.
* Tất cả peer join channel thành công, anchor peer đã định nghĩa.

# LAB 4.2. Creating blockchain identities for the banks

# LAB 4.3. Building the corporate remittance contract

# LAB 4.4. Setting up the IPFS network

# LAB 4.5. Setting up the bank databases

# **LAB 4.**6**.** Building the bank backend servers

# **LAB 4.**7**.** Building the transaction listeners for the banks

# **LAB 4.**8**.** Creating the corporate remittance app frontend in React