Blockchain-Based Asset Management System using Hyperledger Fabric

This project is a complete, end-to-end blockchain application built for a financial institution, using Hyperledger Fabric. It provides a secure, transparent, and immutable ledger for tracking and managing financial accounts.

The system is composed of a Hyperledger Fabric network, a smart contract (chaincode) written in Go, and a containerized REST API for client interaction, fulfilling all requirements of the internship assignment.

Table of Contents

Problem Statement

A financial institution requires a blockchain solution to manage and track assets representing client accounts. The system must ensure the security, transparency, and immutability of these asset records.

The core functionalities required are:

- Asset creation
- Updating asset values
- Querying the world state to read assets
- Retrieving the transaction history for an asset

Each asset represents an account with the following attributes:

- DEALERID

- MSISDN
- MPIN
- BALANCE
- STATUS
- TRANSAMOUNT
- TRANSTYPE
- REMARKS

Project Architecture

The project consists of three main components as specified in the assignment levels:

1. **Hyperledger Fabric Network**: A test network configured with two organizations (Org1, Org2) and a Raft ordering service. This provides the distributed ledger infrastructure.
2. **Go Smart Contract (Chaincode)**: The business logic of the application, deployed on the Fabric network. It defines the functions that can interact with the ledger, such as CreateAsset, ReadAsset, and GetAssetHistory.
3. **Go REST API**: A client-facing web server that exposes simple HTTP endpoints to invoke the smart contract. It uses the Fabric Gateway SDK to communicate with the blockchain and is containerized using Docker.

Technologies Used

- **Blockchain**: Hyperledger Fabric v2.5
- **Programming Language**: Go (for both the chaincode and the REST API)
- **Containerization**: Docker & Docker Compose
- **Environment**: Ubuntu / Windows Subsystem for Linux (WSL2)

Prerequisites

Before you begin, ensure you have the following installed on your system:

- Docker and Docker Compose
- Go programming language (version 1.21 or higher)
- Git
- curl (for testing the API)

Step-by-Step Installation and Execution Guide

The application must be started in a specific order: first the blockchain network, and then the API server. This requires **at least two separate terminals**.

Step 1: Environment Setup

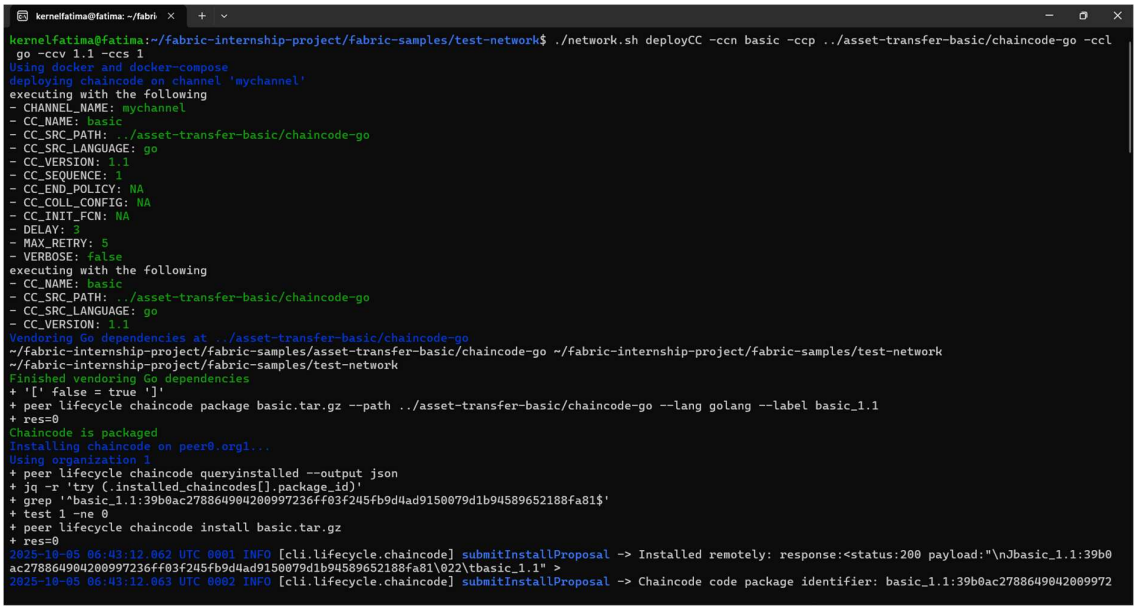
1. Clone this repository to your local machine.
2. # Replace with your actual repository URL
3. git clone [https://github.com/kernelfatima/ Hyperledger-Fabric-Management-System.git]
4. cd fabric-internship-project
5. The fabric-samples directory contains the scripts needed to run the test network. Download the required Hyperledger Fabric binaries and Docker images.
6. cd fabric-samples/test-network
7. ./network.sh prereq

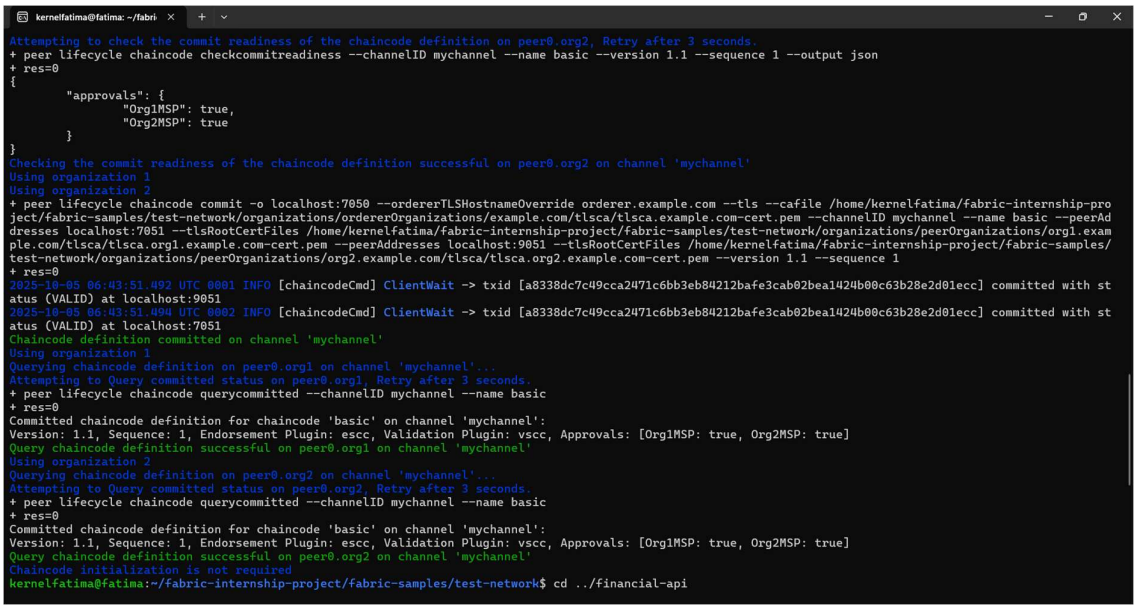
Step 2: Start the Blockchain Network (Terminal 1)

1. Open your **first terminal** window.
2. Navigate to the test-network directory:
3. cd ~/fabric-internship-project/fabric-samples/test-network
4. Bring down any old network instances to ensure a clean start:
5. ./network.sh down
6. Start the Fabric network and create the channel using Certificate Authorities.
7. ./network.sh up createChannel -ca

Step 3: Deploy the Smart Contract (Terminal 1)

1. In the same terminal, deploy the Go smart contract. Since this is a fresh network, the sequence number must be 1.
2. ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go -ccv 1.0 -ccs 1





Leave this terminal open. It is now running your live blockchain network.

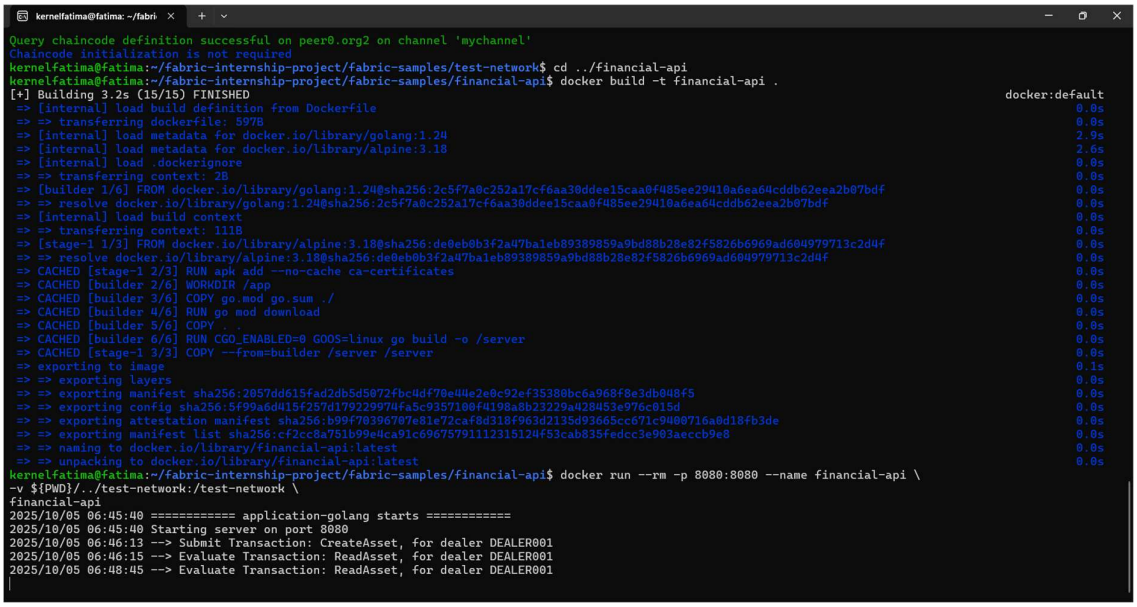
Step 4: Configure and Start the REST API (Terminal 2)

1. Open a **new, second terminal** window.
2. **Important:** The API code needs the IP address of your WSL instance to connect to the Fabric peer. Find your IP address by running:
3. ip addr show eth0 | grep "inet\s" | awk '{print $2}' | cut -d/ -f1

Copy the IP address it returns (e.g., 172.25.1.38).

4. Navigate to the financial-api directory and open the main.go file to update the endpoint.

5. cd ~/fabric-internship-project/fabric-samples/financial-api
6. nano main.go
7. In the editor, find the peerEndpoint constant and

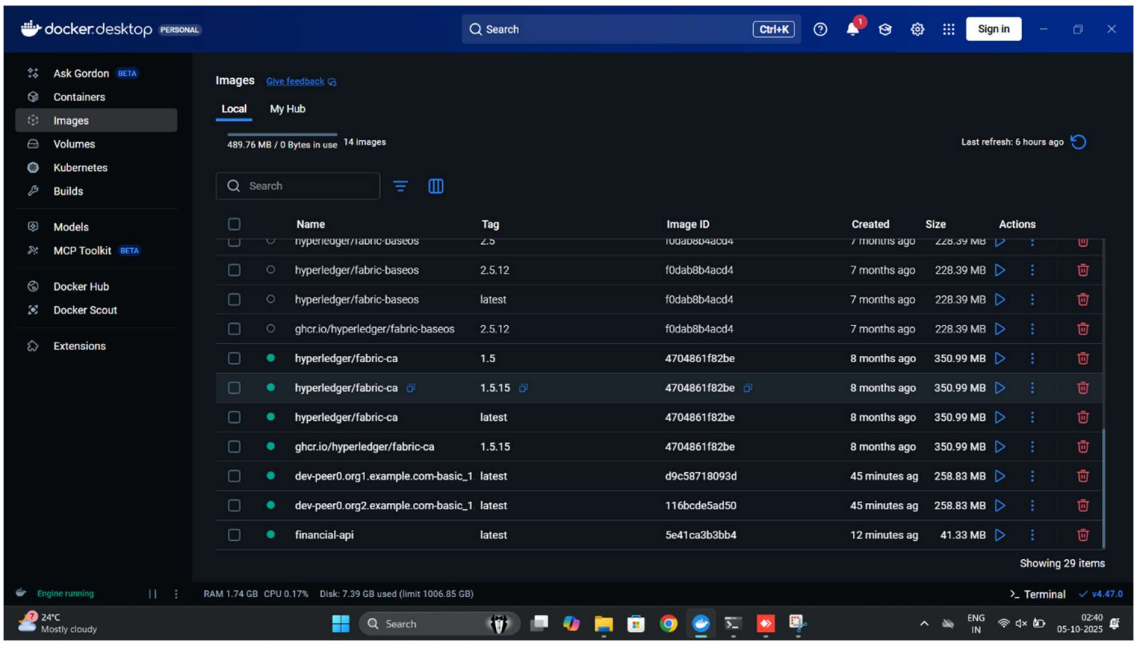


Replace the placeholder with your actual IP address.

8. // Example:
9. peerEndpoint = "172.25.1.38:7051"

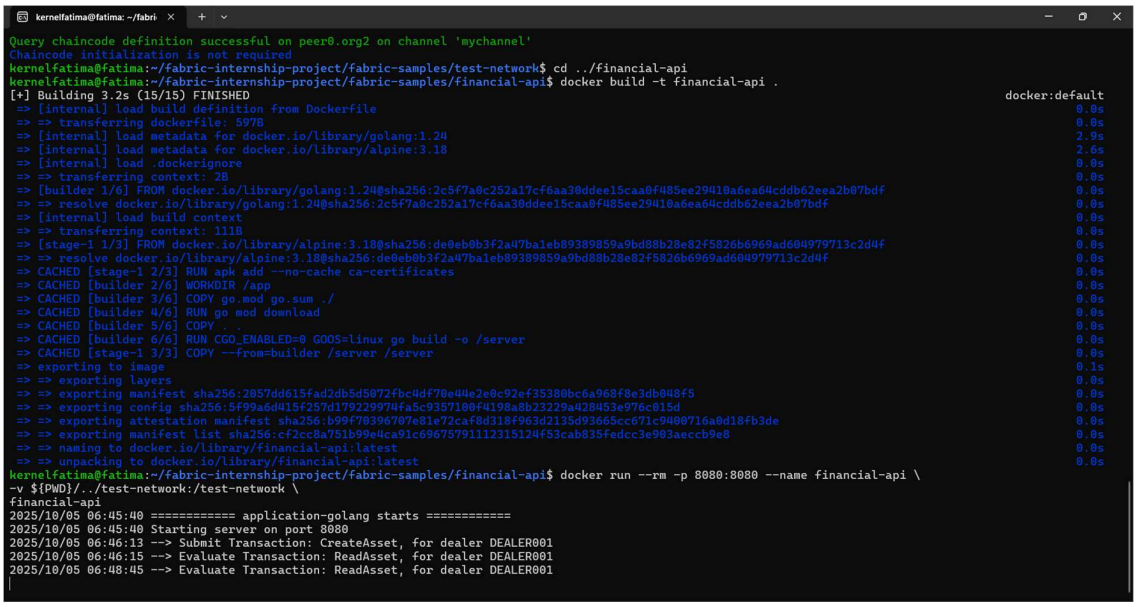
Save and exit the editor (Ctrl + X, Y, Enter).

10. Build the Docker image for the API.



11. docker build -t financial-api .
12. Run the API container.
13. docker run --rm -p 8080:8080 --name financial-api \
14. -v ${PWD}/../test-network:/test-network \
15. financial-api

You should see the log Starting server on port 8080. **Leave this terminal open.** It is now running your live API server.



Step 5: Test the Live Application (Terminal 3)

1. Open a **third, new terminal**.
2. Use curl to interact with your running API.

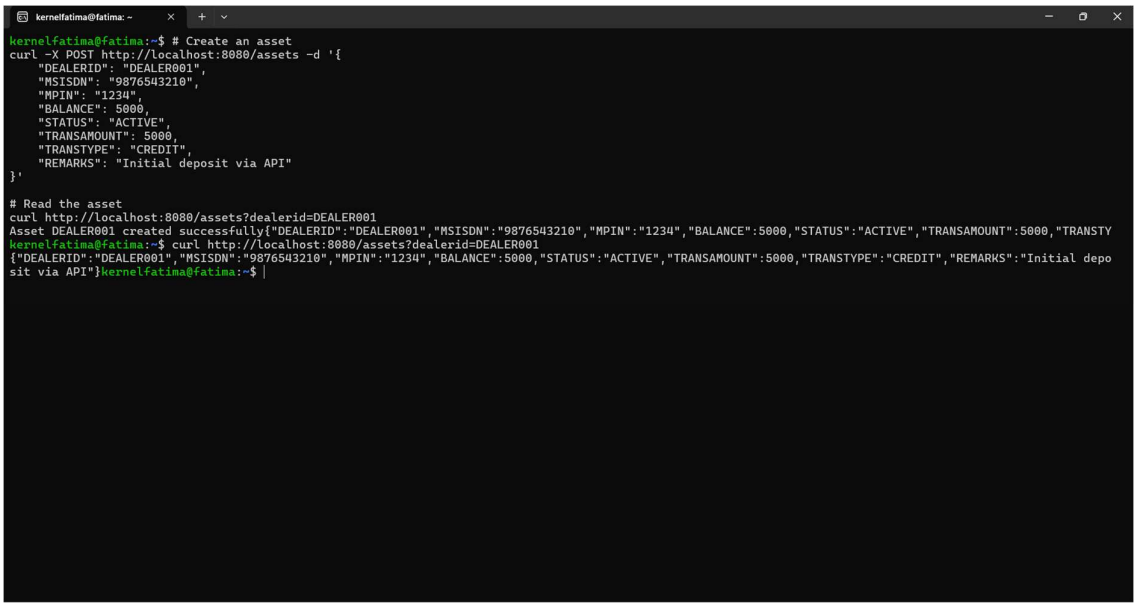
Create an Asset

- **Endpoint:** POST /assets
- **Command:**
- curl -X POST http://localhost:8080/assets -d '{
- "DEALERID": "DEALER001",
- "MSISDN": "9876543210",
- "MPIN": "1234",
- "BALANCE": 5000,

- "STATUS": "ACTIVE",
- "TRANSAMOUNT": 5000,
- "TRANSTYPE": "CREDIT",
- "REMARKS": "Initial deposit via API"
- }'
- **Success Response:** Asset DEALER001 created successfully

Read the Asset

- **Endpoint:** GET /assets?dealerid=<id>
- **Command:**
- curl http://localhost:8080/assets?dealerid=DEALER001
- **Success Response (JSON):**
- {"DEALERID":"DEALER001","MSISDN":"9876543210","MPIN":"1234","BALANCE":5000,"STATUS":"ACTIVE","TRANSAMOUNT":5000,"TRANSTYPE":"CREDIT","REMARKS":"Initial deposit via API"}



Bringing Down the Application

1. In **Terminal 2** (running the API), press Ctrl + C to stop the API server container.
2. In **Terminal 1** (running the blockchain), run the down script to stop and remove all network components.
3. ./network.sh down