



Hyperledger Fabric: Design, Develop and Deploy (LFS270)

Gain a deep understanding of Hyperledger Fabric, including how to write smart contracts, manage Chaincode, handle private data; and create Node.js client apps that interact with Fabric networks, controlling access by user identity.

Duration: 30 Hours

Prerequisites for this course

To best benefit from this course, you should have:

- Knowledge of basic Linux system administration commands and navigation
- Knowledge of bash basics
- Strong knowledge of containerization and Docker
- Familiarity with NoSQL databases and general understanding of CouchDB
- Ability to read JavaScript, TypeScript, and Go programming languages
- Familiarity with YAML

Outline for this course

Chapter 1 – Course Introduction

- Chapter 2 Introduction to Blockchain
- Chapter 3 Introduction to Linux Foundation Decentralized Trust
- Chapter 4 Introduction to Hyperledger Fabric
- Chapter 5 The Hyperledger Fabric Model
- Chapter 6 Identity, Certificate Authority and Membership Service Provider
- Chapter 7 Hyperledger Fabric Components (Peer, Orderer, Channel, and Ledger)
- Chapter 8 The Ordering Service & Its Implementations Using Raft or BFT
- Chapter 9 Transaction Lifecycle
- Chapter 10 Smart Contracts, Chaincode & Fabric Chaincode Lifecycle
- Chapter 11 Hyperledger Fabric Prerequisite Installation and Test
- Chapter 12 Installing and Testing a Fabric Network
- Chapter 13 Network Configuration of Peer and Orderer Nodes





- Chapter 14 Channel Configuration
- Chapter 15 Smart Contract and Chaincode Design, Development, and Deployment
- Chapter 16 Chaincode Interaction Using the CLI
- Chapter 17 Using CouchDB as State Database
- Chapter 18 JSON Rich Query Implementation
- Chapter 19 Fabric CA Implementation
- Chapter 20 Understand and Implement the Fabric Gateway Service
- Chapter 21 Hyperledger Fabric Production Deployment
- Chapter 22 Security & Performance Considerations
- Chapter 23 Writing State-Based Endorsement Policies in Smart Contracts
- Chapter 24 Private Data Collection Implementation