# coding-bootcamps.com

## Hyperledger Fabric for System Admin Vs Developers

Coding Bootcamps

By Matt Zand
from Coding Bootcamps

# Hyperledger Fabric Certifications and Career Choices

Webinar

# Recap

How much do you know about Hyperledger?

## About Presenter

- **Matt** is a serial entrepreneur and the founder of 3 successful tech startups: DC Web Makers, Coding Bootcamps and High School Technology Services.
- He is a leading author of Hands-on Smart Contract Development with Hyperledger Fabric book by O'Reilly Media.
- He has written more than 100 technical articles and tutorials on blockchain development for Hyperledger, Ethereum and Corda platforms.
- He has a master's degree in business management from the University of Maryland.
- Prior to blockchain development and consulting, he worked as senior web and mobile App developer and consultant, angel investor, business advisor for a few startup companies.

# Outline

- Hyperledger Fabric Highlights
- HF Components and Architecture
- HF for System Administration
- HF for Application Development

1. Permissioned architecture
2. Highly modular
3. Pluggable consensus
4. Open smart contract model
5. Low latency of finality/confirmation
6. Flexible approach to data privacy

7. Multi-language smart contract support: Go, Java, JavaScript
8. Support for Ethereum Virtual Machine and Solidity
9. Designed for continuous operations, including rolling upgrades and asymmetric version support
10. Governance and versioning of smart contracts
11. Flexible endorsement model for achieving consensus across required organizations
12. Queryable data (key-based queries and JSON queries)

1. Peer
2. Ordering service
3. Fabric CA (Certificate Authority)
4. Fabric ledger
5. Channel
6. Smart Contracts or chaincodes
7. Endorsement policy
8. Membership services provider (MSP)

# Peer

- Fundamental unit of the HF network
- Peer maintains a ledger and a smart contract
- Peer Types
  - Anchor Peer
  - Leader Peer
  - Endorsing Peer
  - Committing peer

# Peer

- Peers can be created, started, stopped, reconfigured, and even deleted.
- They expose a set of APIs that enable administrators and applications to interact with the services that they provide.

# Ordering Service

- Sometimes also referred as Orderer
- The mechanism by which applications and peers interact with each other to ensure that every peer's ledger is kept consistent with each other is mediated by special nodes called ***orderers***
- Orderer Types
  - Solo (deprecated in v2.x)
  - Kafka (deprecated in v2.x)
  - Raft (recommended)

# Ordering Service

- Orderer atomically broadcasts messages containing transactions to peers to be committed. All peers receive the same block of transactions in the same order.
- The ordering service orders transactions on a first come, first serve basis for all channels on the network.
- After the transaction is ordered, the records of the committed are grouped and assigned as part of the block for that communication channel.

# Ordering Service- Raft versus Kafka

- **Raft is easier to set up**
- **Kafka and Zookeeper are not designed to be run across large networks**.
- **Raft is supported natively**, which means that users are required to get the requisite images and learn how to use Kafka and ZooKeeper on their own.

# Ordering Service- Raft versus Kafka

- Whereas Kafka uses a pool of servers (called “Kafka brokers”) and the admin of the orderer organization specifies how many nodes they want to use on a particular channel, Raft allows the users to specify which ordering nodes will be deployed to which channel.

- Raft is the first step toward Fabric’s development of a **Byzantine Fault Tolerant** (BFT) ordering service.

# Fabric Certificate Authority

- **Fabric CA** is a private root CA provider capable of managing digital identities of Fabric participants that have the form of X.509 certificates.

- Because Fabric CA is a custom CA targeting the Root CA needs of Fabric, it is inherently not capable of providing SSL certificates for general/automatic use in browsers.

# Fabric Certificate Authority

- Custom implementation of certificate authority
- Alternative to cryprogen
- Helps in
  - Registration of identity
  - Enrollment

# Ledger

- It stores important factual information about business objects
- Ledger consist of two parts, world state and blockchain which is nothing but transaction log
- World State has two options
  - LevelDB (Default)
  - CouchDB

# Ledger

- The ledger is the sequenced, tamper-resistant record of all state transitions in the Fabric while State transitions are a result of chaincode invocations (‘transactions’) submitted by participating parties.

- Each transaction results in a set of asset **key-value pairs** that are committed to the ledger as creates, updates, or deletes.

# Ledger Features

- Query and update ledger using key-based lookups, range queries, and composite key queries
- Read-only queries using a rich query language (if using CouchDB as state database)
- Read-only history queries — Query ledger history for a key, enabling data provenance scenarios

# Ledger Features

- Transactions consist of the versions of keys/values that were read in chaincode (read set) and keys/values that were written in chaincode (write set)
- Transactions contain signatures of every endorsing peer and are submitted to ordering service
- Transactions are ordered into blocks and are “delivered” from an ordering service to peers on a channel

# Ledger Features

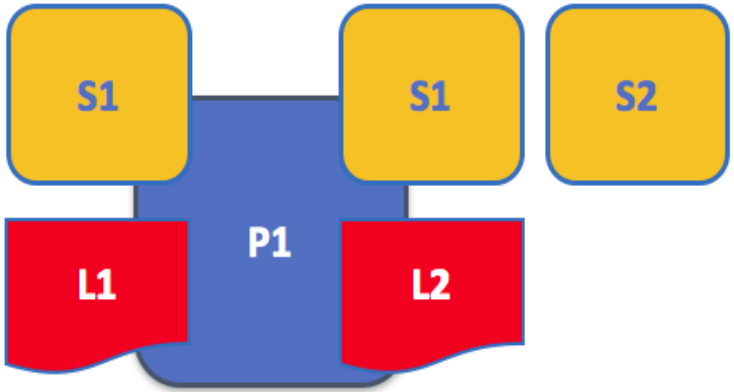
- Peers validate transactions against endorsement policies and enforce the policies
- Prior to appending a block, a versioning check is performed to ensure that states for assets that were read have not changed since chaincode execution time
- There is immutability once a transaction is validated and committed

# Ledger Features

- A channel's ledger contains a configuration block defining policies, access control lists, and other pertinent information

- Channels contain Membership Service Provider instances allowing for crypto materials to be derived from different certificate authorities

# Multiple Ledgers



# Channel

- Mechanism by which interested peers within a blockchain network can collaborate & communicate.
- A Hyperledger Fabric channel is a private “subnet” of communication between two or more specific network members, for the purpose of conducting private and confidential transactions.
- In HF, we have System Channel & Application Channel

# Channel Characteristics

- A channel is defined by members (organizations), anchor peers per member, the shared ledger, chaincode application(s) and the ordering service node(s).

- Each transaction on the network is executed on a channel, where each party must be authenticated and authorized to transact on that channel.

- Each peer that joins a channel, has its own identity given by a membership services provider (MSP), which authenticates each peer to its channel peers and services.

# Channel Characteristics

- The election of a leading peer for each member on a channel determines which peer communicates with the ordering service on behalf of the member.

- **No ledger data can pass from one channel to another**. This separation of ledgers, by channel, is defined and implemented by configuration chaincode, the identity membership service and the gossip data dissemination protocol.

# Smart Contracts

- Smart contracts are nothing but business agreement encapsulated in the code.

- Smart Contract vs Chaincode
  - Smart contracts are meant to contain the logic that gets executed and interacts with the ledger in order to manage and maintain the state of the asset.

  - Chaincode governs the administrative aspects of smart contracts that is how smart contracts are packaged in a chaincode and then deployed on the network.

# Smart Contracts…

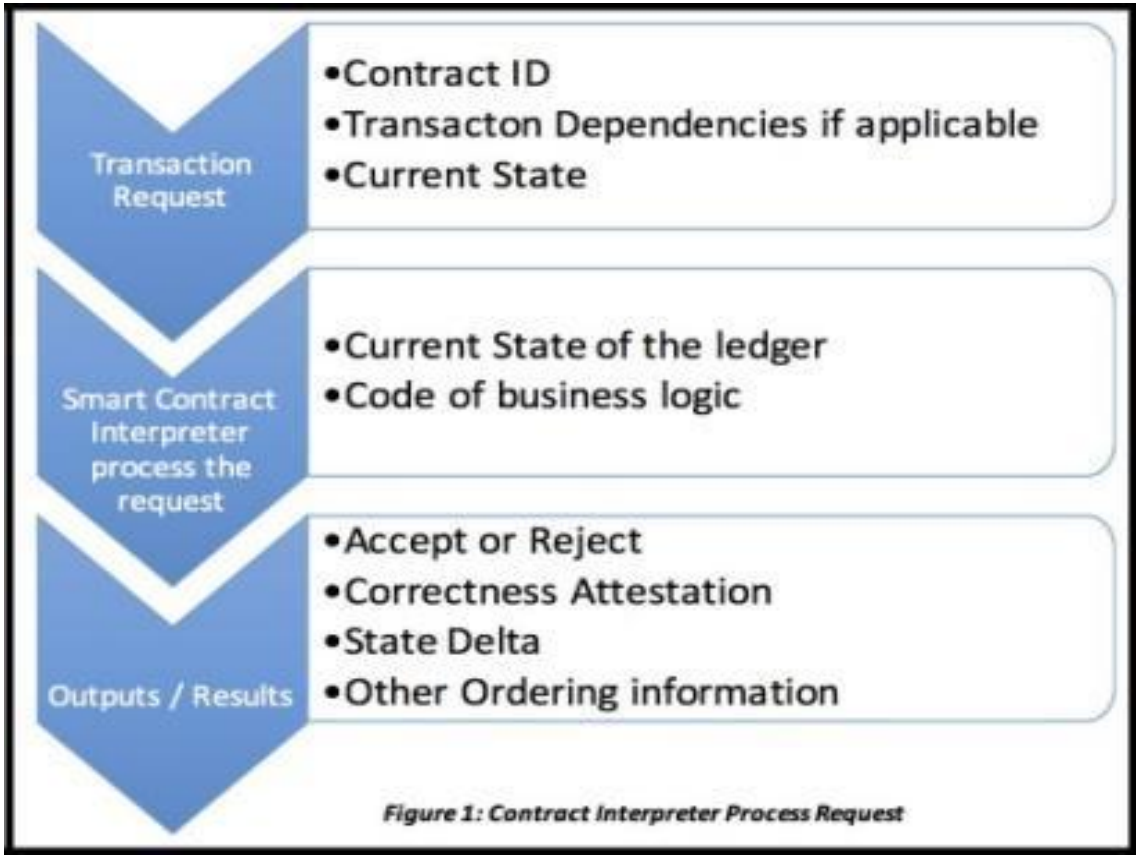
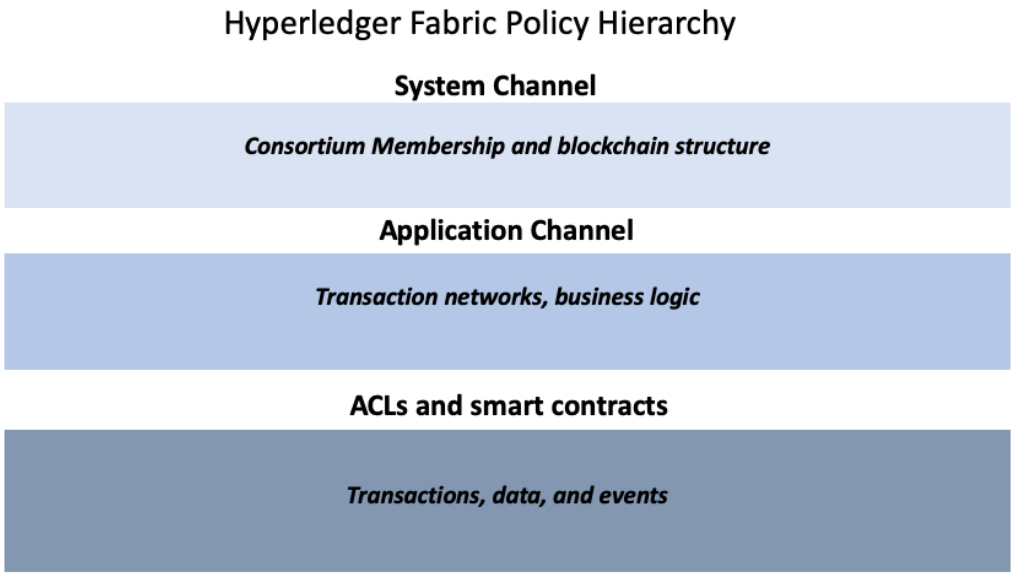


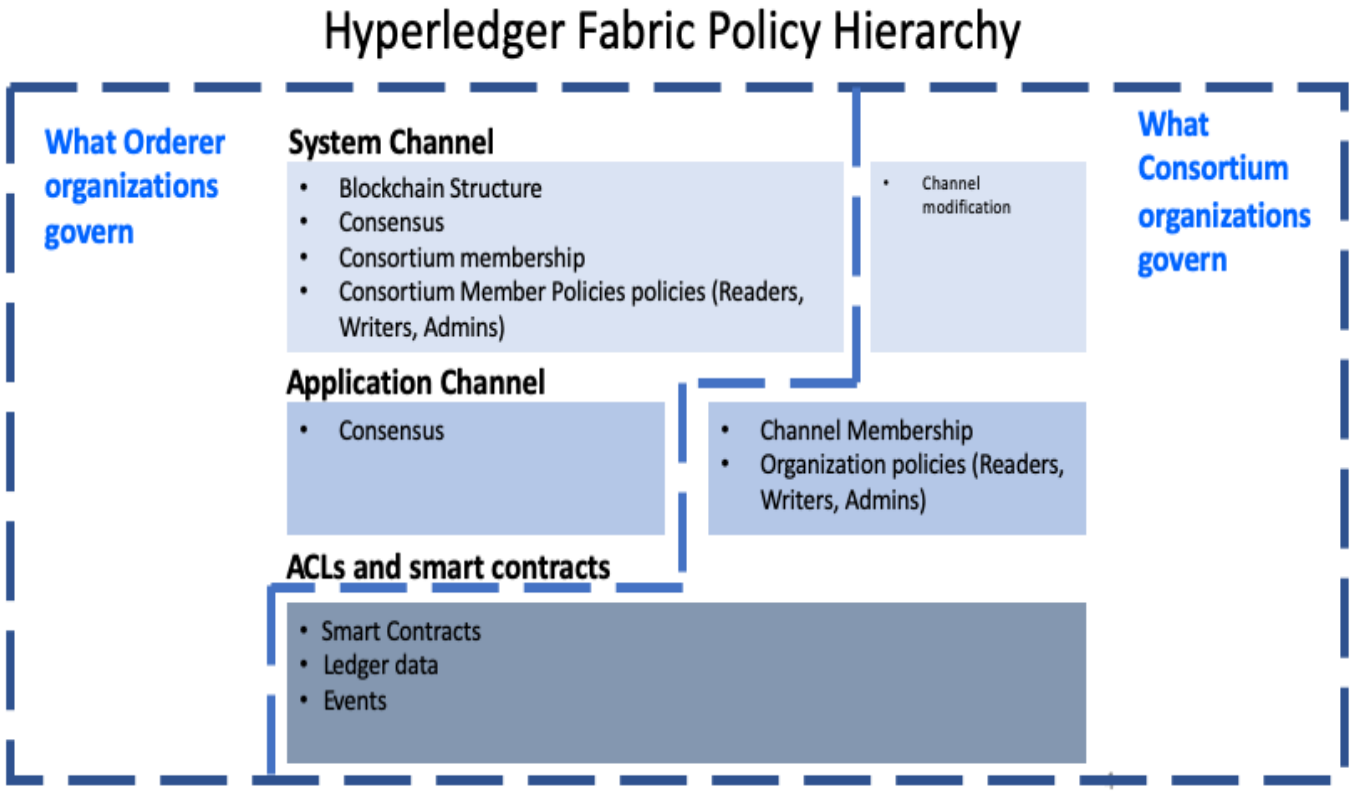

*Figure 1: Contract Interpreter Process Request*

# Fabric Policies

Policies are implemented at different levels of a Fabric network. Each policy domain governs different aspects of how a network operates.



# Fabric Policies



# Endorsement Policy

- Define the smallest set of organizations that are required to endorse a transaction in order for it to be valid.
- Described or set during chaincode instantiation.
- Principals are described as 'MSP.ROLE', where MSP represents the required MSP ID and ROLE represents one of the four accepted roles: **member**, **admin**, **client**, and **peer**.

# Endorsement Policy Syntax

**Here are a few examples of valid principals:**

- 'Org0.admin': any administrator of the Org0 MSP
- 'Org1.member': any member of the Org1 MSP
- 'Org1.client': any client of the Org1 MSP
- 'Org1.peer': any peer of the Org1 MSP

The syntax of the language is: **EXPR(E[, E...])**

**Where EXPR is either AND, OR, or OutOf, and E is either a principal (with the syntax described above) or another nested call to EXPR.**

# Membership Service Provider (MSP)

- A mechanism that provides members with a set of roles and permissions within the network.

- Implementation of the MSP requirement is a set of folders that are added

- MSP Types
  - Local MSP
  - Channel MSP

# Membership Service Provider (MSP)

- **Certificate Authorities** issue identities by generating a public and private key which forms a key-pair that can be used to prove identity.

- Because a private key can never be shared publicly, a mechanism is required to enable that proof which is where the MSP comes in. For example, a peer uses its private key to digitally sign, or endorse, a transaction.

- **the MSP is the mechanism that allows that identity to be trusted and recognized by the rest of the network without ever revealing the member's private key**

# Membership Service Provider (MSP)

- **Whereas Certificate Authorities generate the certificates that represent identities, the MSP contains a list of permissioned identities.**

- Also, it is the MSP that turns an identity into a **role** by identifying specific privileges an actor has on a node or channel.

- **Note that when a user is registered with a Fabric CA, a role of admin, peer, client, orderer, or member must be associated with the user.**

- The role of system admin is to:

  Administer and interact with chaincode, manage peers, and operate basic CA-level functions.

  Being a HF system admin entails a good understanding of the HF network topology, chaincode operations, administration of identities, permissions, how and where to configure component logging, and much more.

- System admin’s main tasks are:

- Application Lifecycle Management
- Install and Configure Network
- Diagnostics and Troubleshooting including logs
- Membership Service Provider
- Network Maintenance and Operations

- System admin technical requirements:

From technical standpoint, to become a professional HF system admin, one should acquire a basic knowledge of the Linux command line, bash, Docker and containers (like Kubernetes), NoSQL, CouchDB, and blockchain and distributed ledgers.

- Certification exam for system admins:

Certified Hyperledger Fabric Administrator
(CHFA) certification is highly recommended.

- The role of Fabric Developer is to:

  In comparison with system administration, HF developers' primary task is to work on one major component of HF which is smart contract or chaincode. In short, developers are required to design, develop, test and deploy HF chaincodes on the peer.

- Fabric developer's main tasks are:

- Define transaction functions
- Execute simple queries
- Create complex queries
- Define assets using key value pairs
- Identify potentially private data
- Incorporate private data collection
- Submit, evaluate, and query transaction by invoking the smart contracts

- Technical requirements for Fabric developers:

HF developers must have a solid foundation or background in one of the following programming languages: JavaScript, Java, Go, or Python. The most famous JS framework that is currently used by Fabric developers is Node.JS.

- Certification exam for Fabric developers:

Certified Hyperledger Fabric Developer certification is highly recommended.

# Recap

What we learned in this webinar?

# Resources- Hyperledger Training

Hyperledger Fabric for System Admins

Hyperledger Fabric for Developers

Intro to Hyperledger Sawtooth for System Admins

# Resources- Articles

- Comprehensive Blockchain Hyperledger Developer Guide from Beginner to Advance Level

- How to Install Hyperledger Fabric on AWS

- Hyperledger Fabric for System Administers versus Developers

- Best practices for managing enterprise blockchain networks on Hyperledger

# Resources- Articles…

- How to Add Organizations to Channel in Hyperledger Fabric
- How to Build Blockchain Applications in Hyperledger Fabric
- How to Build Hyperledger Fabric via Inventory Asset Management Application
- How to Compile and Deploy Hyperledger Fabric Chaincode
- How to Develop Application with Hyperledger Fabric through SDK

# Q/A

# coding-bootcamps.com

Thank you

Coding
Bootcamps