**Lecture 4.7 Notes**

**Hyperledger Fabric**

**7.1 Hyperledger**

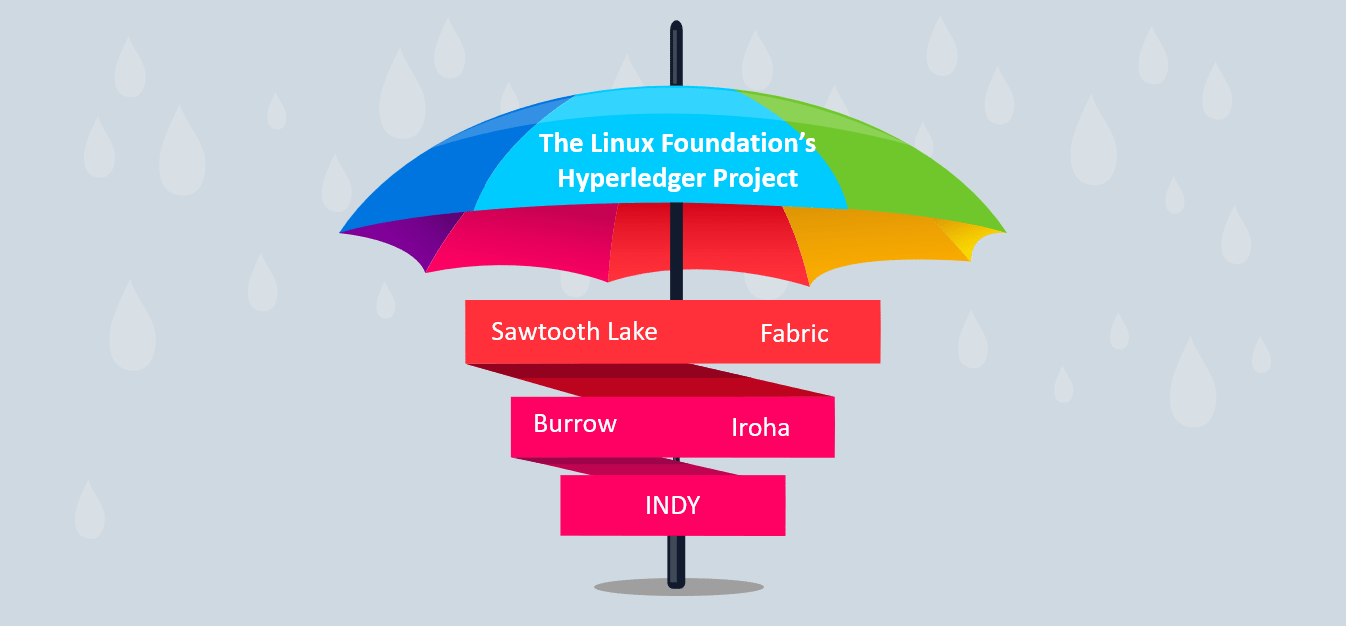
* Hyperledger is an open source community focused on developing a suite of stable frameworks, tools and libraries for enterprise-grade blockchain deployments.
* It serves as a neutral home for various distributed ledger frameworks including Hyperledger Fabric, Sawtooth, Indy, as well as tools like Hyperledger Caliper and libraries like Hyperledger Ursa.

## **Hyperledger Project**

As stated on the Hyperledger’s website,

“Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, Internet of Things, supply chains, manufacturing, and Technology.”

Hyperledger incubates a plethora of business blockchain technologies, framework, under its “**Umbrella strategy**“. Currently, Hyperledger houses the following projects:



* As the popularity of Bitcoin, Ethereum and a few other derivative technologies grew, interest in applying the underlying technology of the blockchain, distributed ledger and distributed application platform to more innovative *enterprise* use cases also grew.
* However, many enterprise use cases require performance characteristics that the permissionless blockchain technologies are unable (presently) to deliver.
* In addition, in many use cases, the identity of the participants is a hard requirement, such as in the case of financial transactions where Know-Your-Customer (KYC) and Anti-Money Laundering (AML) regulations must be followed.

For enterprise use, we need to consider the following requirements:

* Participants must be identified/identifiable
* Networks need to be *permissioned*
* High transaction throughput performance
* Low latency of transaction confirmation
* Privacy and confidentiality of transactions and data pertaining to business transactions

While many early blockchain platforms are currently being *adapted* for enterprise use, Hyperledger Fabric has been *designed* for enterprise use from the outset.

**7.2 Hyperledger Fabric**

* Hyperledger Fabric is an open source enterprise-grade permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts, that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms.
* One key point of differentiation is that Hyperledger was established under the Linux Foundation, which itself has a long and very successful history of nurturing open source projects under **open governance** that grow strong sustaining communities and thriving ecosystems.
* Hyperledger is governed by a diverse technical steering committee, and the Hyperledger Fabric project by a diverse set of maintainers from multiple organizations.
* It has a development community that has grown to over 35 organizations and nearly 200 developers since its earliest commits.
* Fabric has a highly **modular** and **configurable** architecture, enabling innovation, versatility and optimization for a broad range of industry use cases including banking, finance, insurance, healthcare, human resources, supply chain and even digital music delivery.
* Fabric is the first distributed ledger platform to support **smart contracts authored in general-purpose programming languages** such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL).
* This means that most enterprises already have the skill set needed to develop smart contracts, and no additional training to learn a new language or DSL is needed.
* The Fabric platform is also **permissioned**, meaning that, unlike with a public permissionless network, the participants are known to each other, rather than anonymous and therefore fully untrusted.
* This means that while the participants may not *fully* trust one another (they may, for example, be competitors in the same industry), a network can be operated under a governance model that is built off of what trust *does* exist between participants, such as a legal agreement or framework for handling disputes.
* One of the most important of the platform’s differentiators is its support for **pluggable consensus protocols** that enable the platform to be more effectively customized to fit particular use cases and trust models.
* For instance, when deployed within a single enterprise, or operated by a trusted authority, fully byzantine fault tolerant consensus might be considered unnecessary and an excessive drag on performance and throughput. In situations such as that, a crash fault-tolerant (CFT) consensus protocol might be more than adequate whereas, in a multi-party, decentralized use case, a more traditional byzantine fault tolerant (BFT) consensus protocol might be required.

# **7.3 Hyperledger Fabric Model**

This section outlines the key design features woven into Hyperledger Fabric that fulfill its promise of a comprehensive, yet customizable, enterprise blockchain solution:

## **Assets**

* Assets can range from the tangible (real estate and hardware) to the intangible (contracts and intellectual property).
* Hyperledger Fabric provides the ability to modify assets using chaincode transactions.
* Assets are represented in Hyperledger Fabric as a collection of key-value pairs, with state changes recorded as transactions on a Channel ledger.
* Assets can be represented in binary and/or JSON form.

## **Chaincode**

* Chaincode is software defining an asset or assets, and the transaction instructions for modifying the asset(s); in other words, it’s the business logic.
* Chaincode enforces the rules for reading or altering key-value pairs or other state database information.
* Chaincode functions execute against the ledger’s current state database and are initiated through a transaction proposal.
* Chaincode execution results in a set of key-value writes (write set) that can be submitted to the network and applied to the ledger on all peers.

## **Ledger Features**

The ledger is the sequenced, tamper-resistant record of all state transitions in the fabric. 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.

The ledger is comprised of a blockchain (‘chain’) to store the immutable, sequenced record in blocks, as well as a state database to maintain current fabric state. There is one ledger per channel. Each peer maintains a copy of the ledger for each channel of which they are a member.

Some features of a Fabric ledger:

* 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
* 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
* 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
* 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

## **Privacy**

* Hyperledger Fabric employs an immutable ledger on a per-channel basis, as well as chaincode that can manipulate and modify the current state of assets (i.e. update key-value pairs).
* A ledger exists in the scope of a channel, it can be shared across the entire network (assuming every participant is operating on one common channel) or it can be privatized to include only a specific set of participants.
* In the latter scenario, these participants would create a separate channel and thereby isolate/segregate their transactions and ledger.
* In order to solve scenarios that want to bridge the gap between total transparency and privacy, chaincode can be installed only on peers that need to access the asset states to perform reads and writes (in other words, if a chaincode is not installed on a peer, it will not be able to properly interface with the ledger).
* When a subset of organizations on that channel need to keep their transaction data confidential, a private data collection (collection) is used to segregate this data in a private database, logically separate from the channel ledger, accessible only to the authorized subset of organizations.
* Thus, channels keep transactions private from the broader network whereas collections keep data private between subsets of organizations on the channel.
* To further obfuscate the data, values within chaincode can be encrypted (in part or in total) using common cryptographic algorithms such as AES before sending transactions to the ordering service and appending blocks to the ledger.
* Once encrypted data has been written to the ledger, it can be decrypted only by a user in possession of the corresponding key that was used to generate the cipher text.

## **Security & Membership Services**

* Hyperledger Fabric underpins a transactional network where all participants have known identities. Public Key Infrastructure is used to generate cryptographic certificates which are tied to organizations, network components, and end users or client applications.
* As a result, data access control can be manipulated and governed on the broader network and on channel levels. This “permissioned” notion of Hyperledger Fabric, coupled with the existence and capabilities of channels, helps address scenarios where privacy and confidentiality are paramount concerns.

## **Consensus**

* In distributed ledger technology, consensus has recently become synonymous with a specific algorithm, within a single function.
* However, consensus encompasses more than simply agreeing upon the order of transactions, and this differentiation is highlighted in Hyperledger Fabric through its fundamental role in the entire transaction flow, from proposal and endorsement, to ordering, validation and commitment.
* In a nutshell, consensus is defined as the full-circle verification of the correctness of a set of transactions comprising a block.
* Consensus is achieved ultimately when the order and results of a block’s transactions have met the explicit policy criteria checks.
* These checks and balances take place during the lifecycle of a transaction, and include the usage of endorsement policies to dictate which specific members must endorse a certain transaction class, as well as system chaincodes to ensure that these policies are enforced and upheld.
* Prior to commitment, the peers will employ these system chaincodes to make sure that enough endorsements are present, and that they were derived from the appropriate entities.
* Moreover, a versioning check will take place during which the current state of the ledger is agreed or consented upon, before any blocks containing transactions are appended to the ledger.
* This final check provides protection against double spend operations and other threats that might compromise data integrity, and allows for functions to be executed against non-static variables.
* In addition to the multitude of endorsement, validity and versioning checks that take place, there are also ongoing identity verifications happening in all directions of the transaction flow.
* Access control lists are implemented on hierarchical layers of the network (ordering service down to channels), and payloads are repeatedly signed, verified and authenticated as a transaction proposal passes through the different architectural components.
* To conclude, consensus is not merely limited to the agreed upon order of a batch of transactions; rather, it is an overarching characterization that is achieved as a byproduct of the ongoing verifications that take place during a transaction’s journey from proposal to commitment.

**7.4 Transaction Flow**

* Ethereum (a public, permissionless blockchain) and Quorum (private, permissioned blockchain based on Ethereum code) are based on execute-order architecture. Some of the limitations that this introduces are sequential execution of all transaction which directly affects transaction throughput.
* The main concept that differentiates Hyperledger Fabric from other blockchains is its execute-order-validate architecture.
* Transactions in Hyperledger Fabric do need not be executed by each peer.
* We can define the endorsement policy that specifies which peer nodes have to execute the transaction and give their endorsement.
* This means that we can define a subset of peers to execute (endorse) a given transaction and satisfy the transaction’s endorsement policy.
* Therefore, this allows for parallel execution of transactions and directly boosts performances of the system. Hyperledger separates transaction flow in three distinct steps:

1. **Transaction Execution** – running the smart contract code
2. **Ordering** through a consensus protocol
3. **Transaction Validation**



* The transaction flow is the way in which consensus is achieved.
* The client application is going to submit the transaction to a few peers and these peers going to execute this transaction and agree that the output is the same across all of them. There are all going to find the output of the transaction and they will add their signature to it as a part of endorsement.
* The client application must collect endorsements from multiple peers in the networks to say that this transaction is the valid transaction, and all the outputs are the same.
* Once if collected sufficient signatures then it can submit the transactions for ordering, now the ordering service will ensure that all of them are fully ordered and totally ordered across all the nodes, Once you determine the order of all the transactions then the resolution of validation.

**7.5 Key Benefits of Hyperledger Fabric**

